(1)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SONAR BASED NAVIGATION OF AN AUTONOMOUS UNDERWATER VEHICLE

by

Alp KAYIRHAN

June 1994

Thesis Advisor:	Roberto Cristi
Second Reader:	Harold A.Titus

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis
4. SONAR BASED NAVIGATION OF AN AUTONOMOUS UNDERWATER VEHICLE UNCLASSIFIED	5. FUNDING NUMBERS	
6. AUTHOR(S) Alp Kayirhan		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORIN G AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A	

13. ABSTRACT *(maximum 200 words)*

A navigation algorithm to navigate an AUV within a charted environment is presented. The algorithm uses sonar range measurements and incorporates them with a potential function which defines the map of the operation area. Extended Kalman filtering is used in the algorithm. Least squares techniques are used in the estimation of system parameters. The algorithm is tested by both computer generated data and actual data collected from the vehicle NPS AUVII during tests in a water tank. Fixed interval smoothing is applied in order to smooth the estimates produced by the Kalman filter. The effects of currents in the operation area are sought. An approach based on backpropagation neural networks for the navigation algorithm is also presented. Throughout the simulation studies the algorithm yields a robust and reliable solution to the navigation problem of AUV's.

14. SUBJECT TERMS AUV, Kalman Filter, Extended Kalman Filter, Fixed Interval Smoothing, ARX Model, Least Squares Estimate, Potential Function, Neural Networks, Backpropagation Adaptive Learning, Momentum			15. NUMBER OF PAGES 93
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Sonar Based Navigation of an Autonomous Underwater Vehicle

by

Alp Kayirhan
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE (EE)

from the

NAVAL POSTGRADUATE SCHOOL
June 1994

Author: [illegible]
Alp Kayirhan

Approved by: [illegible]
Roberto Cristi, Thesis Advisor

[illegible]
Harold A. Titus, Second Reader

[illegible]
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

A navigation algorithm to navigate an AUV within a charted environment is presented. The algorithm uses sonar range measurements and incorporates them with a potential function which defines the [illegible] of the operation area. Extended Kalman filtering is used in the algorithm. Least squares techniques are used in the estimation of system parameters. The algorithm is tested by both computer generated data and actual data collected from the vehicle NPS AUVII during tests in a water tank. Fixed interval smoothing is applied in order to smooth the estimates produced by the Kalman filter. The effects of currents in the operation area are sought. An approach based on backpropagation neural networks for the navigation algorithm is also presented. Throughout the simulation studies the algorithm yields a robust and reliable solution to the navigation problem of AUV's.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I would like to thank my advisor, Roberto Cristi for his guidance and encouragement throughout this thesis, his expertise and through knowledge made this thesis a great learning experience.

I would also like to thank my wife Yasemin for her unyielding support and love.

I. INTRODUCTION

A. GENERAL

Since the early 1960s the U.S Navy has been interested in Unmanned Underwater Vehicles(UUV), which include Autonomous Underwater Vehicles (AUV) and Remotely Operated Vehicles(ROV). The latter are tethered to a supporting platform, where an operator provides necessary control signals. In this case most of the mission planning and on-line decision making is accomplished by the operator. Increasing computer capabilities have decreased the need to tether the vehicle to prove guidance and navigational support. The improvements allowed underwater vehicles to be autonomous. [Ref. 1]

Autonomous vehicles can go where humans cannot go and do not want to go. Autonomous vehicles are capable of receiving initial input, moving to another location, executing a mission, and returning with the requested results and the data. In addition to performing labor intensive and repetitive tasks, these vehicles can perform their tasks faster and with greater precision than humans, and can also proceed into hostile or contaminated environments. [Ref. 2]

At the Naval Postgraduate School the interest in autonomous underwater vehicles has manifested itself in current research which supports development of a prototype vehicle named NPS AUVII.

B. GOAL

If an autonomous underwater vehicle is to be completely autonomous, it must have a navigation system capable of estimating its own position at all times. The effort in this thesis is mainly concerned with the navigation problem of the AUV. Compared to autonomous land vehicles, AUVs have many disadvantages, such as the presence of highly nonlinear vehicle dynamics that are often uncertain and the presence of currents whose effects cannot be detected by inertial navigation systems. One possible solution to this problem is the use of velocity measurements relative to the ocean floor, using, for example, a doppler sonar. However, the cost, size and the depth of the ocean often prevents use of doppler sonar on standard AUV's. Visual information (sonar and optical) seems a viable approach in environments having landmarks as references, such as buoys, piers, rocks, and man made structures.

C. METHOD OF APPROACH

This thesis develops a short range navigation algorithm for the planar motion of the NPS AUVII vehicle. The system uses sonar returns and incorporates them with a potential function which defines the map of the operation area. The measurements obtained from sonar are filtered through a Kalman estimator using extended Kalman filtering. The main feature of the algorithm is the use of a potential function to define the area of operation. By use of this potential function, the system discriminates the objects which are not on the map , and yields a robust and reliable solution to the short range navigation problem of the AUV. The algorithm is first applied in a rectangular shaped pool and later applied to an environment without any borders (e.g open sea).

This thesis consists of five parts. Chapter II discusses the theory behind the navigator design and the modeling assumptions in this study. Chapter III discusses the details of the potential function approach in the implementation of a navigation algorithm together with simulation results. The results using the actual data collected in the water tank are also discussed. Chapter IV discusses the implementation of neural networks for the navigation of AUV. Consequently Chapter V summarizes the results and conclusions of the study, and provides recommendations for further investigation.

II. BACKGROUND

A. KALMAN FILTERING

Since it was derived by R.E. Kalman in 1960, the Kalman filter has been used extensively in the design of optimal estimation algorithms. Its rapid acceptance and subsequent success are due to the Kalman Filter's recursive nature and optimality.

The Kalman filter estimates the states of a system given a set of inputs to the system and a set of measurements both corrupted by noise [Ref. 3]. The system is assumed to be driven by both a known input and a random input:

$$x(k+1)=\Phi x(k)+\Delta_1 u(k)+\Delta_2 w(k)\,, \tag{2.1}$$

where u(k) is the deterministic input, w(k) is plant driving noise, Φ is the state transition matrix and x(k) is the state vector. We also assume w(k) is a zero mean gaussian random vector with covariance:

$$E[w(k)\,w(k+n)]=Q\delta(n)\,, \tag{2.2}$$

where Q is the covariance matrix of plant noise and $\delta(n)$ is impulse function. The measurement process can be modeled as:

$$y(k)=Hx(k)+v(k)\,, \tag{2.3}$$

where v(k) is zero mean additive gaussian noise and H is the measurement matrix. Measurement noise v(k) has covariance:

$$E[v(k)\,v(k+n)]=R\delta(n)\,, \tag{2.4}$$

where R is the measurement error covariance matrix.

Using the orthogonality principle it can be shown that the recursive solution to this problem has the following form:

$$\hat{x}(k+1|k+1)=\hat{x}(k+1|k)+G(k+1)\,[y(k+1)-\hat{y}(k+1|k)]\,. \tag{2.5}$$

Given the current estimate we can predict ahead by simply using the state equation:

$$\hat{x}(k+1|k)=\phi\hat{x}(k|k)+\Delta_1 u(k)\,, \tag{2.6}$$

which yields the optimal predictor. We can reasonably estimate $\hat{y}$(k+1) by simply using the H matrix such that:

$$\hat{y}(k+1|k)=H\hat{x}(k+1|k)\,. \tag{2.7}$$

The Kalman gain G depends on the covariance matrix of estimation error which is defined as:

$$P(k|k) = E[\{x(k)-\hat{x}(k|k)\}\{x(k)-\hat{x}(k|k)\}^T]\,. \tag{2.8}$$

The Kalman gains G(k+1) are found by applying the orthogonality principle as shown in [Ref. 3], which leads to the following recursive equations:

$$G(k+1) = P(k+1|k)H^T[HPH^T + R]^{-1} \tag{2.9}$$

$$P(k+1|k+1) = [I - G(k+1)H]P(k+1|k) \tag{2.10}$$

$$P(k+1|k) = \phi\hat{x}(k|k)\phi^T + \Delta_2 Q \Delta_2{}^T \tag{2.11}$$

The filter equations can be summed up as follows:

$$\hat{x}(k+1|k) = \phi\hat{x}(k|k) + \Delta_1 u(k) \tag{2.12}$$

$$\hat{y}(k+1|k) = H\hat{x}(k+1|k) \tag{2.13}$$

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + G(k+1)[y(k+1) - \hat{y}(k+1|k)] \tag{2.14}$$

As can be seen from the above equations, for a Linear Time Invariant (LTI) system, the Kalman gains can be computed off-line and stored in the computer and used as look-up tables. In order to start the filter algorithm, a *priori* values of the state estimates and the error covariance are needed. The proper choice for these estimates are:

$$\hat{x}(0|0) = E[x(0)], \tag{2.15}$$

$$P(0|0) = Cov[x(0)]. \tag{2.16}$$

The Q matrix is the covariance of the state excitation noise. If the disturbances to the system are uncorrelated, the Q matrix is diagonal. The Q matrix accounts for the noise processes in the system as well as modeling errors between model and system dynamics.

R matrix is the covariance matrix of measurement error. Large values of R means that the measurements are not consistent with the dynamic model.

The matrices Q and R represent a trade-off between the model and the observation accuracies. In particular if the entries of R are larger than the entries of Q, it means that the model is more reliable than the observations, and the Kalman Filter will give less emphasis to the observation. Vice versa, if the entries of R are small compared to Q , it means that the observations are more reliable than the model, and the Kalman filter puts more emphasis on the observations.

While the matrix R is in general determined by sensing device, the determination of Q is more difficult due to its lack of physical meaning, and it is usually set by trial and error.

As mentioned above for the implementation of the filter algorithm we need a *priori* information for both the states and the error covariance. Large values in the initial estimate of P means that the filter will not solely depend on the initial conditions and gives more emphasis to the measurements.

B. EXTENDED KALMAN FILTERING

The Kalman filter yields a statistically optimal estimate for the states of a LTI system. Unfortunately many real systems such as the model studied in this thesis are nonlinear in nature. A general nonlinear system driven by both a deterministic and random input is:

$$x(k+1)=f(x(k),u(k),w(k),k), \tag{2.17}$$

where w(k) is a random forcing function with covariance Q, u(k) is a deterministic input and f(.) is a nonlinear function. The measurement process can be modeled as:

$$y(k)=g(x(k),v(k),k), \tag{2.18}$$

where v(k) is measurement noise with covariance R and g(.) is a nonlinear function.

Our task is to find the linear estimate of x(k), which minimizes the mean square estimation error. Solution to this problem is given by the conditional expectation. But generating and maximizing the conditional expectation is a formidable task which is not suitable to on-line implementations.

A simpler and of course mostly used approach is to use an extension of the Kalman filter by using Taylor series expansion. The resulting filter is suboptimal and does not give a guaranteed convergence as the Kalman filter does. In spite of this,the Extended Kalman Filter (EKF) is used in a wide range of applications, especially in target tracking systems where the nonlinearity stems form the geometry involved in the measurement process.

The extended Kalman filter equations are similar to the Kalman equations:

$$\hat{x}(k+1|k)=f(\hat{x}(k|k),u(k),0,k), \tag{2.19}$$

$$\hat{y}(k+1|k)=g(\hat{x}(k+1|k),0,k), \tag{2.20}$$

$$\hat{x}(k+1|k+1)=\hat{x}(k+1|k)+G(k+1)\,[y(k+1)-\hat{y}(k+1|k)] \tag{2.21}$$

As can be seen from these equations the states are predicted ahead using the estimated current state and the known input. Since the noise is unknown, it is set to its expected value of zero. The Kalman gains are computed by using the first order linear approximation of the nonlinear model using Taylor series expansion. These approximations are as follows:

$$\Phi=\frac{\partial f(x(k),u(k),w(k),k)}{\partial x(k)}\Big|_{\substack{x(k)=\hat{x}(k|k)\\ u(k)=u(k)\\ w(k)=0}} \tag{2.22}$$

$$\Delta=\frac{\partial f(x(k),u(k),w(k),k)}{\partial u(k)}\Big|_{\substack{x(k)=\hat{x}(k|k)\\ u(k)=u(k)\\ w(k)=0}} \tag{2.23}$$

$$H=\frac{\partial g(x(k),v(k),k)}{\partial x(k)}\Big|_{\substack{x(k)=\hat{x}(k|k)\\ v(k)=0}} \tag{2.24}$$

$$D=\frac{\partial g(x(k),v(k),k)}{\partial v(k)}\Big|_{\substack{x(k)=\hat{x}(k|k)\\ v(k)=0}}. \tag{2.25}$$

These linearized values are used in the gain equations:

$$P(k+1|k)=\phi P(k|k)\phi^{T}+\Delta Q\Delta^{T} \qquad (2.26)$$

$$G(k+1)=P(k+1|k)+H^{T}[HPH^{T}+DRD^{T}]^{-1} \qquad (2.27)$$

$$P(k+1|k+1)=[I-G(k+1)H]P(k+1|k) \qquad (2.28)$$

It is easy to see, from these equations, that the Kalma gains should be computed on-line since the linearized model depends on the current state estimate.

C. SMOOTHING ALGORITHM

Smoothing is a procedure that uses all the state estimates and associated error covariances produced by the Kalman Filter and attempts to improve the accuracy of these estimates.

Let k be within the time interval 0 to N, that is, $0 \le k \le N$, for some fixed N. The Kalman filter estimate at time k denoted by $\hat{x}(k+1|k)$ is based only on the measurements up to time k. The smoothed estimate is based on the measurements that occurred over the entire time interval. The smoothed estimate is denoted by $\hat{x}(k|N)$ and the smoothed error covariance is given by $P(k|N)$.

Meditch, [Ref. 4], classifies the smoothing algorithms into three categories:

Fixed interval smoothing, denoted by $\hat{x}(k|N)$ where $k=0,1..N-1$; N is a positive integer.

Fixed point smoothing, denoted by $\hat{x}(k|j)$ where $j=k+1,k+2,..k$ and k is a fixed integer.

Fixed lag smoothing, denoted by $\hat{x}(k|k+L)$, where $k=0,1,..N$, is a positive integer, and L is the fixed time lag.

In this study we use a fixed interval smoothing algorithm to smooth the estimates of the extended Kalman filter. The equations used in the smoothing algorithm are given below:

$$A(k)=P(k+1|k+1)\phi^T P^{-1}(k+1|k) \qquad (2.29)$$

$$\hat{x}(k|N) = \hat{x}(k+1|k+1) + A(k)\,[\hat{x}(k+1|N)-\hat{x}(k+1|k)] \qquad (2.30)$$

$$P(k|N) = P(k+1|k+1)+A(k)\,[P(k+1|N)-P(k+1|k)]\,A(k)^T \qquad (2.31)$$

As can be seen from the above equations, the smoothed estimate is the Kalman filter estimate $\hat{x}(k+1|k+1)$, adjusted by a weighted error term. This error is the difference between the smoothed estimate and the predicted estimate calculated by the Kalman Filter. The smoothed error covariance has no impact in the smoothing algorithm but it is a measure of how well the smoothing filter is working. If $P(k|N) \le P(k+1|k+1)$ this means the smoothed estimate is better than or equal to its filtered estimate.

The fixed interval smoothing algorithm is an off-line procedure that operates backward in time. The smoothing filter is initialized by the last filtered estimate, that is to say the last filtered estimate becomes the first smoothed estimate. It is also clear that, in order to apply the smoothing algorithm, the values of $\hat{x}(k+1|k)$, $\hat{x}(k+1|k+1)$, $P(k+1|k)$, $P(k+1|k+1)$ must be stored in computer prior to fixed interval smoothing.

D. PARAMETER ESTIMATION

In this section we present an algorithm to estimate the system parameters using least squares method. System identification is mainly concerned with fitting a dynamic model to the measured input and output data. Trying all possible dynamic models is certainly a formidable task, therefore we restrict our models to a certain model structure. We consider the following difference equation:

$$A(z^{-1})y(k)=B(z^{-1})u(k)+C(z^{-1})e(k), \qquad (2.32)$$

where e(k) is white noise and y(k) and x(k) are output and input sequences respectively. A, B, C are polynomials in the time delay operator expressed as:

$$\begin{aligned} A(z^{-1}) &= 1+a_1z^{-1}+\ldots\ldots+a_nz^{-n} \\ B(z^{-1}) &= \quad b_1z^{-1}+\ldots\ldots+b_nz^{-n} \qquad (2.33) \\ C(z^{-1}) &= 1+c_1z^{-1}+\ldots\ldots+c_nz^{-n}. \end{aligned}$$

This input output model is called ***ARMAX***, which stands for Auto Regressive Moving Average Extended [Ref. 5]. A particular case of this model is *the* ***ARX*** model which is used in this thesis. It comes from the ***ARMAX*** model with the moving average part removed and expressed as:

$$A(z^{-1})y(k)=B(z^{-1})u(k)+e(k). \tag{2.34}$$

We can write this in regression form as:

$$y(k)=\phi^T(k)\underline{\theta}+e(k), \tag{2.35}$$

where

$$\begin{gathered}\phi^T=[-y(k-1),\ldots,-y(k-n),u(k-1),\ldots,u(k-n)]\\ \underline{\theta}=[a_1\ldots a_n\ b_1\ldots b_n]^T.\end{gathered} \tag{2.36}$$

The basic method to estimate the parameters $\underline{\theta}$ is the least squares method which minimizes the mean square estimation error:

$$\underline{\hat{\theta}}=argmin\sum_{k=1}^{M}|y(k)-\phi^T\underline{\theta}|^2, \tag{2.37}$$

with respect to $\underline{\theta}$. The solution to this problem is given by pseudo inverse matrix:

$$\underline{\hat{\theta}}=(\Phi^T\Phi)^{-1}\Phi^T y, \tag{2.38}$$

where

$$\Phi=[\phi(1),\phi(2),\ldots\ldots\phi(M)]^T$$
$$y=[y(1),y(2),\ldots,y(M)]^T, \tag{2.39}$$

with M being the number of data points used in the parameter estimation.

If we consider a first order difference equation of the form:

$$y(k)+ay(k-1)=bu(k-1)+e(k) \tag{2.40}$$

Then using Equations (2,38) and (2,39) the least squares estimate of the parameters *a* and *b* are given as follows:

$$\begin{vmatrix}\hat{a}\\ \hat{b}\end{vmatrix}=\begin{vmatrix}\sum_{k=1}^{M}y(k-1)^2 & -\sum_{k=1}^{M}y(k-1)u(k-1)\\ -\sum_{k=1}^{M}y(k-1)u(k-1) & \sum_{k=1}^{M}u(k-1)^2\end{vmatrix}^{-1}\begin{vmatrix}-\sum_{k=1}^{M}y(k-1)y(k)\\ \sum_{k=1}^{M}u(k-1)y(k)\end{vmatrix} \tag{2.41}$$

Using either the pseudo inverse matrix form or the summation form, the parameters *a* and *b* can be estimated using the above equations.

E. SYSTEM MODELING

Our study in this thesis is concerned with the planar motion of the AUV, therefore system modeling for the planar motion is considered. The system model used in this study is the same as in [Ref. 6].

In modeling the planar motion of the AUV we use a earth fixed cartesian coordinate frame. We define the states of the vehicle $\mathbf{X} \epsilon \mathbf{R}^5$ as:

$$\mathbf{X(t)} = \begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix}. \tag{2.42}$$

The states x and y define the x and y position of the AUV. The state v is the forward velocity of the AUV and the states θ and $\dot{\theta}$ are the heading(yaw) and heading rate of the vehicle.

The differential equations describing the model are as follows:

$$\begin{gathered} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{v} = -a_1 v + a_2 u_1 \\ \ddot{\theta} = -b_1\dot{\theta} + b_2 u_2, \end{gathered} \tag{2.43}$$

where a_1, a_2, b_1, b_2 are constants depending on the vehicle dynamics and the inputs u_1, u_2 are the RPM and rudder commands to the system respectively.

The system model can now be expressed in state space form as:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & \cos\theta & 0 & 0 \\ 0 & 0 & \sin\theta & 0 & 0 \\ 0 & 0 & -a_1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -b_1 \end{bmatrix} \mathbf{X} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ a_2 & 0 \\ 0 & 0 \\ 0 & b_2 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \mathbf{W}, \qquad (2.44)$$

where **W** is zero mean gaussian noise which accounts for the errors between the model and the actual system dynamics. Notice that this model does not include side-slip since we assume the velocity vector to be in line with the heading of the vehicle. For simplicity, the error due to side slip is taken into account by the noise term **W**.

In the next chapter we are going to use this model in conjunction with the model of the environment to determine the optimal state estimator.

III. APPLICATION OF POTENTIAL FUNCTIONS

A. POTENTIAL FUNCTION THEORY

Autonomous underwater vehicles must have the ability to locate themselves in partially known environments using sensors available to them. In this study, we address the problem of navigating the Autonomous Underwater Vehicle(AUV) using sonar returns. These sonar returns are incorporated with a potential function and used as nonlinear measurements in Kalman Filter. The potential function is a function which, in a sense, defines the area where the vehicle operates. The basic idea behind the potential function is the following:

i) it returns a value of zero at the boundaries of the object and returns a value between one and zero elsewhere;

ii) its derivative is maximum at the boundaries of the object.

As an example, if we consider a pool of rectangular shape with sides L_1 and L_2 we can define a function $\beta(x,y)$ such that:

$$\beta(x,y)=x(x-L_1)y(y-L_2)\,, \tag{3.1}$$

where we take the lower left-hand corner of the pool as the origin of the reference frame. If the sonar return is from the sides of the pool it returns a value of zero. We combine the defining function with a squashing function to provide boundedness and define the potential function as:

$$V(x,y) = F_\lambda(\beta(x,y)). \tag{3.2}$$

The function $F_\lambda(z)$ must be smooth, bounded and differentiable. A possibility is to choose $F_\lambda(z)$ to be a sigmoid function:

$$F_\lambda(z) = \frac{1-e^{-\lambda/z}}{1+e^{-\lambda/z}}, \tag{3.3}$$

with λ being a strictly positive parameter.

The potential function must satisfy the following requirements:

i) V should be continuous and differentiable.

ii) $V(x_0,y_0)$ should return a value of zero if the return is from a reflecting surface.

iii) V should be uniformly bounded.

Note that the first and the third requirements are met by the sigmoid function, while the second is met by the defining function β. The plot of a sigmoid function for various values of λ is given in Figure (3.1). The function is continuous and bounded over the interval [-1, 1] and as the parameter λ decreases, the "sharpness" of the function increases.

If we consider a field of operation containing several obstacles O_1, $O_2, \ldots,$ O_n, each of which has elliptical shape, we can describe a defining function for each obstacle as:

$$\beta_j = [x\ y]\, A_j \begin{bmatrix} x \\ y \end{bmatrix} - C_j, \tag{3.4}$$

where $A_j \in \mathbb{R}^{2x2}$ and $C_j \in \mathbb{R}$ [Ref. 7]. The total defining function is the product of each function:

$$\beta(x,y) = \prod_j \beta_j(x,y) . \tag{3.5}$$

As an example, if we want to add a buoy of cylindrical shape of radius r at point(a,b) in the pool the defining function will be:

$$\beta_{obst}(x,y) = (x-a)^2 + (y-b)^2 - r^2, \tag{3.6}$$

and the total defining function will be:

$$\beta_{tot}(x,y) = x(x-L_1)y(y-L_2)((x-a)^2+(y-b)^2-r^2)), \tag{3.7}$$

where the overall potential function is:

$$V(x,y) = F_\lambda(\beta_{tot}(x,y)) . \tag{3.8}$$

By following the above procedure, additional obstacles may be added to the environment.

The parameter λ has a vital role in shaping the potential function, the small values in λ result in a steeper potential function surface while larger values result in a smoother surface. In Figure (3.2) we see the 3D plot of a potential function in the pool. It is easily seen that at the pool borders it gives a value of zero, whereas at the other points it returns a value between zero and one.

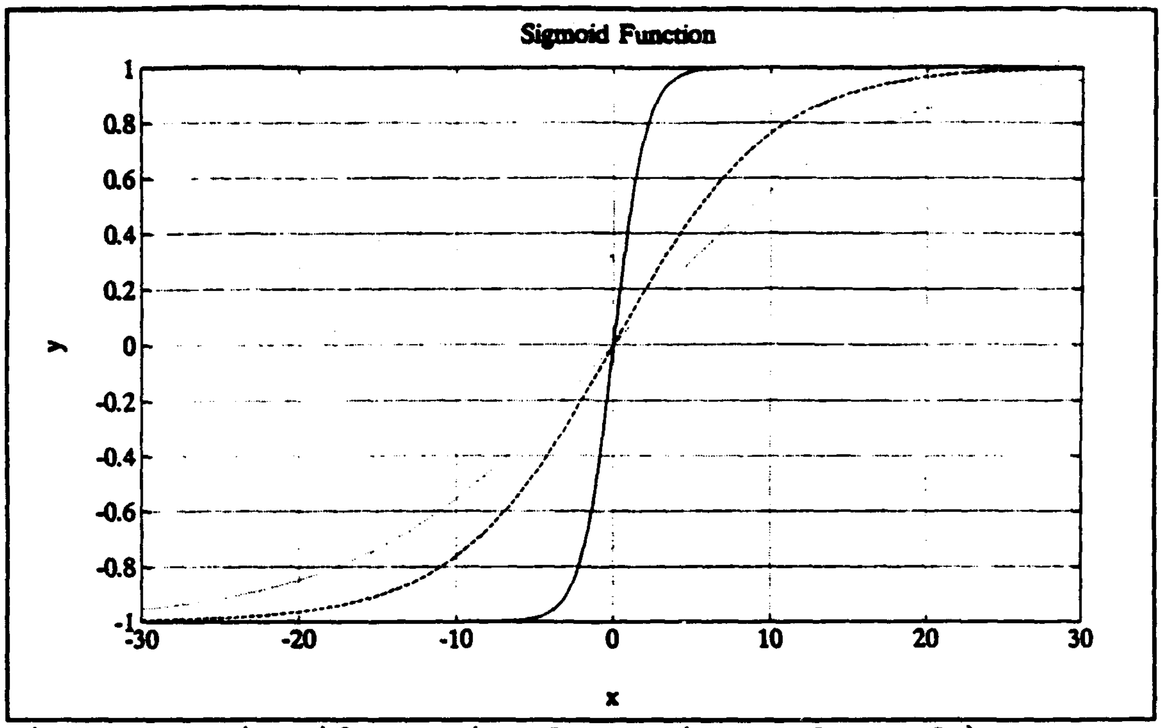

Figure 3.1 Sigmoid Function for various values of λ.

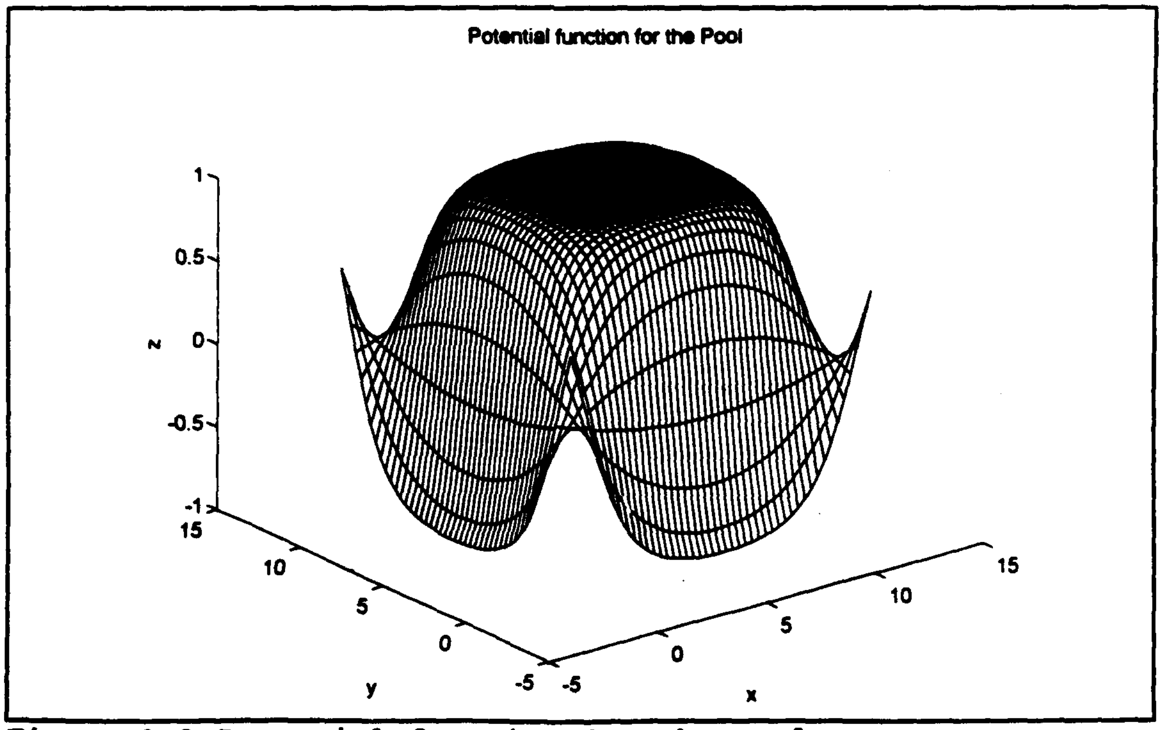

Figure 3.2 Potential function for the pool.

B. APPLICATION TO SYSTEM MODEL

The system model in this study has the form of Equation (2.44). The vehicle states are position in x and y coordinates, velocity, heading(yaw) and heading rate respectively. The navigation algorithm which is based on the extended Kalman Filter uses the sonar range measurements. Therefore we need to define a measurement equation which relates the states of the vehicle to the sonar range measurement. The measurements available to the system are sonar ranges ρ at angles α with respect to the longitudinal axis of the vehicle. We define the location of the "tip" of the sonar range as

$$[x0,y0]=(x+\rho\cos(\theta+\alpha),y+\rho\sin(\theta+\alpha)), \qquad (3.9)$$

where x, y, and θ are the current position and heading of the vehicle.

We relate the sonar range measurement to the system model via the potential function so that the measurement equation becomes:

$$y=g(\boldsymbol{X},\rho,\alpha)=V(x+\rho\cos(\theta+\alpha),y+\rho\sin(\theta+\alpha))+\boldsymbol{W1}, \qquad (3.10)$$

where the function g is a nonlinear function of the states and the sonar measurement and **W1** is zero mean gaussian measurement noise. The function g is the potential function which defines the area of operation. Apart from measurement noise the measurement equation should have the value of zero if the sonar return is consistent with the map of the area.

By defining the above measurement equation we can write the overall model in the standard state space form as:

$$\begin{aligned} \dot{\boldsymbol{X}} &= A\boldsymbol{X} + Bu + \boldsymbol{W} \\ y &= g(X,\rho,\alpha) + \boldsymbol{W1} \end{aligned} \tag{3.11}$$

The system has a nonlinear state equation and a nonlinear observation equation. In order to apply extended Kalman filtering we need to get the linearized observation matrix H as in Equation(2,24), which is:

$$H=\frac{\partial g(\boldsymbol{X},\rho,\alpha)}{\partial \boldsymbol{X}}=[\frac{\partial g}{\partial x},\ \frac{\partial g}{\partial y},\ 0,\ \frac{\partial g}{\partial \theta}\ ,0]\ . \tag{3.12}$$

The partial derivatives are found by application of the chain rule:

$$\begin{gathered} \frac{\partial g}{\partial x}=\frac{\partial F_\lambda}{\partial z}\frac{\partial z}{\partial x} \qquad \frac{\partial g}{\partial y}=\frac{\partial F_\lambda}{\partial z}\frac{\partial z}{\partial y} \\ \frac{\partial g}{\partial \theta}=\frac{\partial F_\lambda}{\partial z}\frac{\partial z}{\partial x}\frac{\partial x}{\partial \theta}+\frac{\partial F_\lambda}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial \theta}, \end{gathered} \tag{3.13}$$

where the function $F_\lambda(z)$ is the sigmoid function as in Equation(3,3).

C. SIMULATIONS IN A RECTANGULAR POOL

The estimation algorithm was tested first in a pool of size 12X12 feet. The Tritech ST725 high resolution sonar onboard NPS AUVII was simulated. The sonar has a rotating head with a scan rate of nine seconds and yields 400 sonar returns per scan corresponding to 0.8 degrees per return. The data from seven successive scans of the sonar head are processed by the Kalman filter. At point (9,9) in the pool we placed a buoy of radius 0.5 feet, and the potential function becomes:

$$V(x,y)=F_{\lambda}([x(x-12)y(y-12)][(x-9)^2+(y-9)^2-0.25]). \quad \textbf{(3.14)}$$

In Figure (3.3) the position estimation with known initial location is shown. The location of the sonar tip, corresponding to the estimated borders of the pool is plotted for each of the seven successive sonar scans and the contour of the potential function is superimposed. In the last plot the estimated trajectory of the vehicle is shown. The estimated trajectory of the vehicle matches the actual trajectory.

The parameter λ plays an important role in the estimation algorithm. Its value should be kept high enough to correct for errors in the initial estimates and low enough to discriminate objects which are not on the map. The filter uses a time varying λ which decreases with time so that the potential function surface get steeper as the estimates converge to the actual values. When there is not enough information in the initial position of the vehicle the filter starts with a large value of λ to account for the errors in the initial estimates.

In Figure (3.4) result of the estimation with unknown initial position is shown. The estimated trajectory converges to the actual trajectory. The effects of the time varying λ can be seen by examining the contours of the potential function. Figure (3.5) shows the result of estimation where we have an error in the initial heading as well as the initial position. The system is more sensitive to heading perturbations, however, the estimation converges to the actual trajectory. Figure (3.6) shows the result of simulation for the case when the obstacle at point (9,9) is not defined in the potential function. The uncharted obstacle is detected by the algorithm and the actual trajectory is recovered from the estimation.

In Figures (3.7) and (3.8) we repeat the experiment with two cylindrical buoys placed at locations (2,2) and (9,9) in the pool. In Figure (3.8) the obstacles are not defined in the potential function. Results of estimation show that the algorithm proves to be robust in the presence of uncharted obstacles.

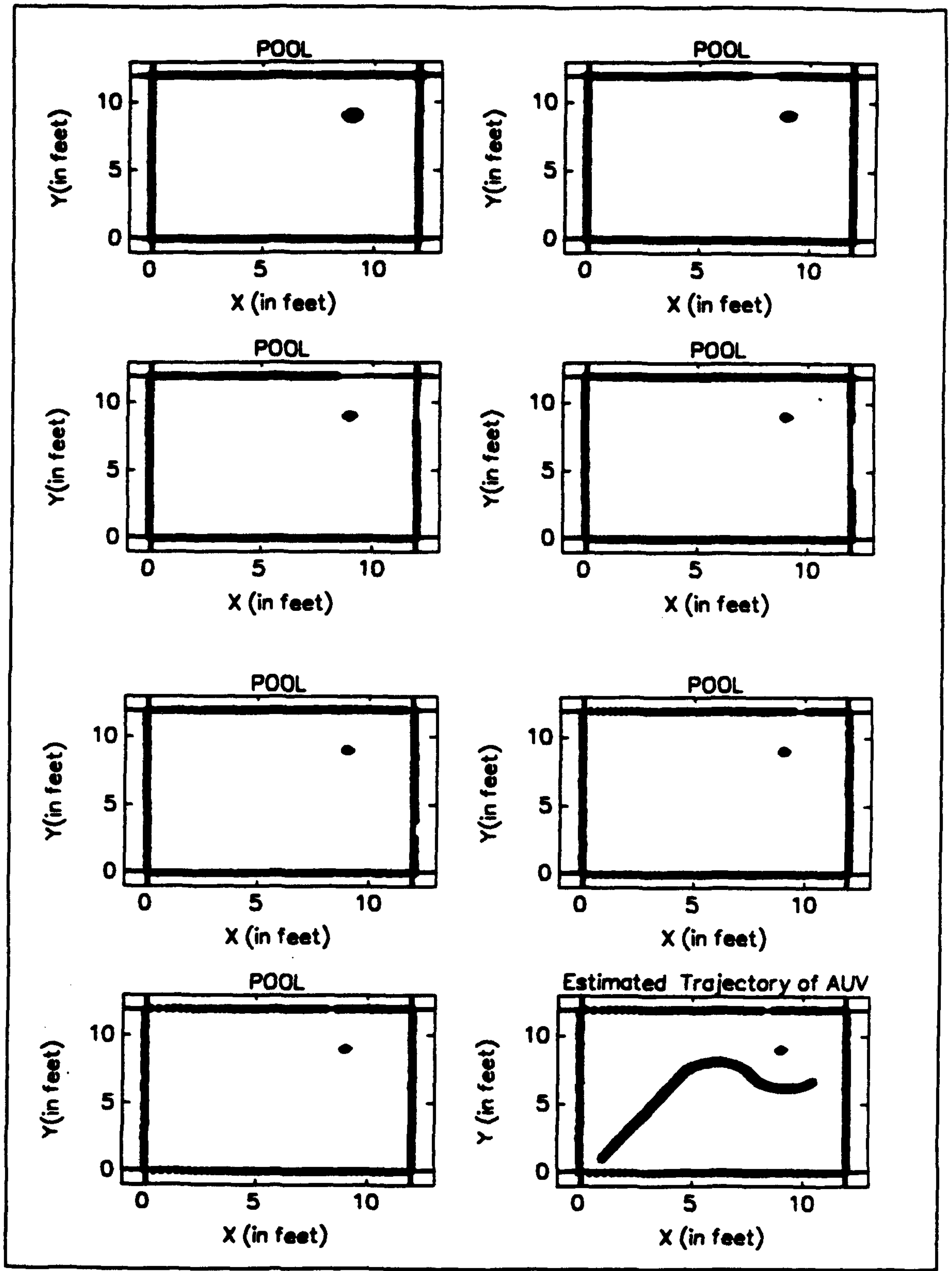

Figure 3.3 Position estimation with known initial location, one obstacle present.

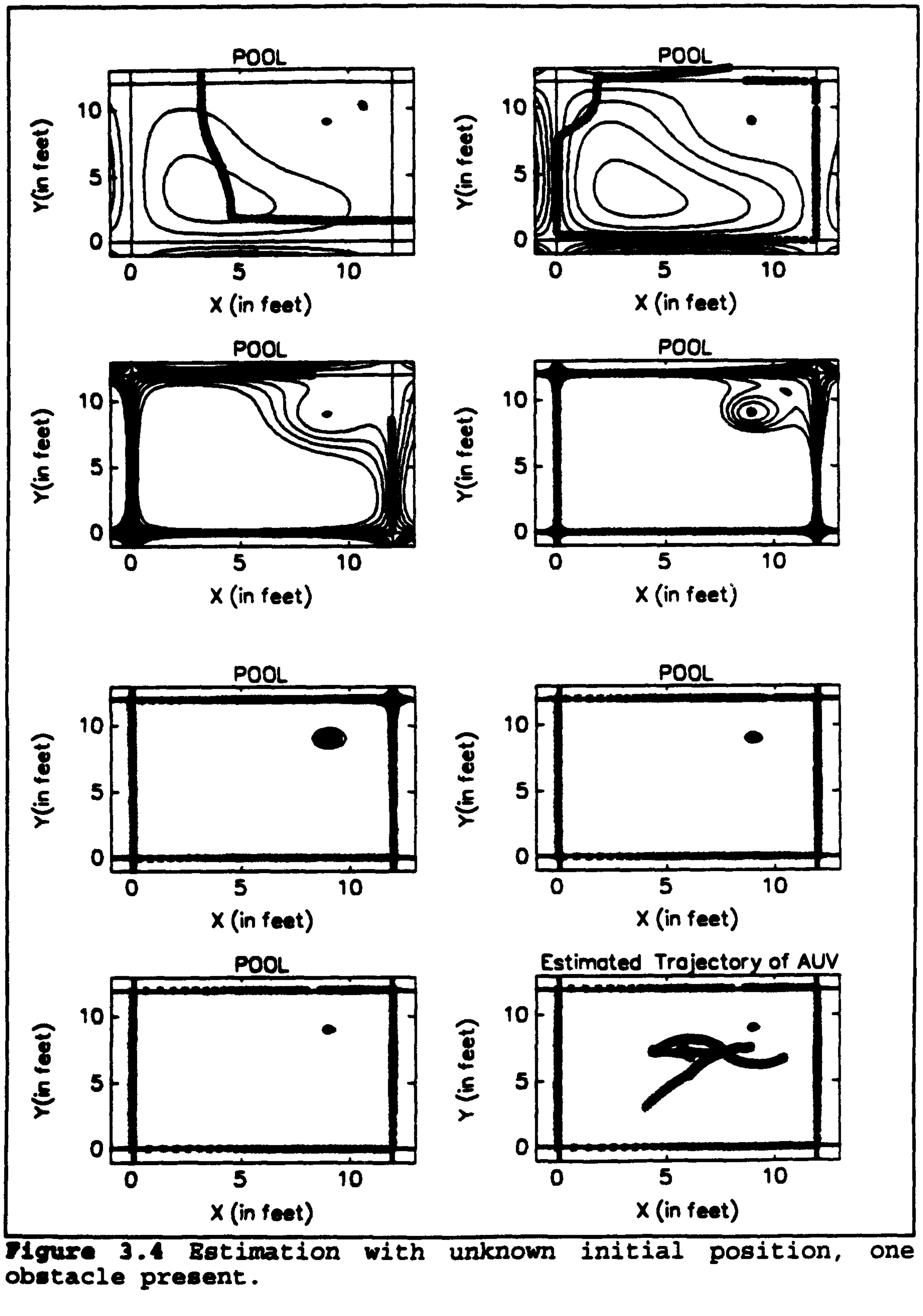

Figure 3.4 Estimation with unknown initial position, one obstacle present.

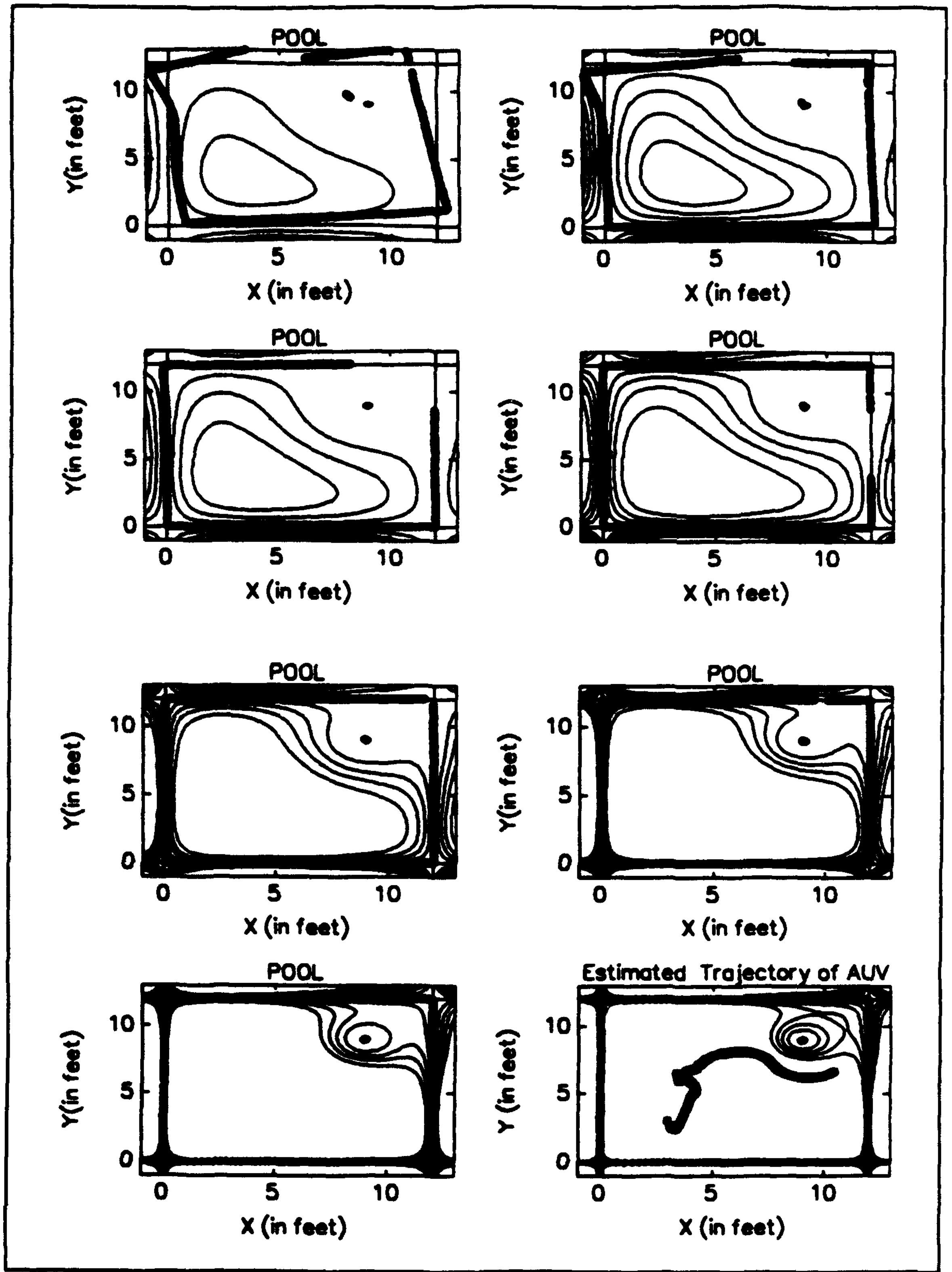

Figure 3.5 Estimation with an error in initial position and heading, one obstacle present.

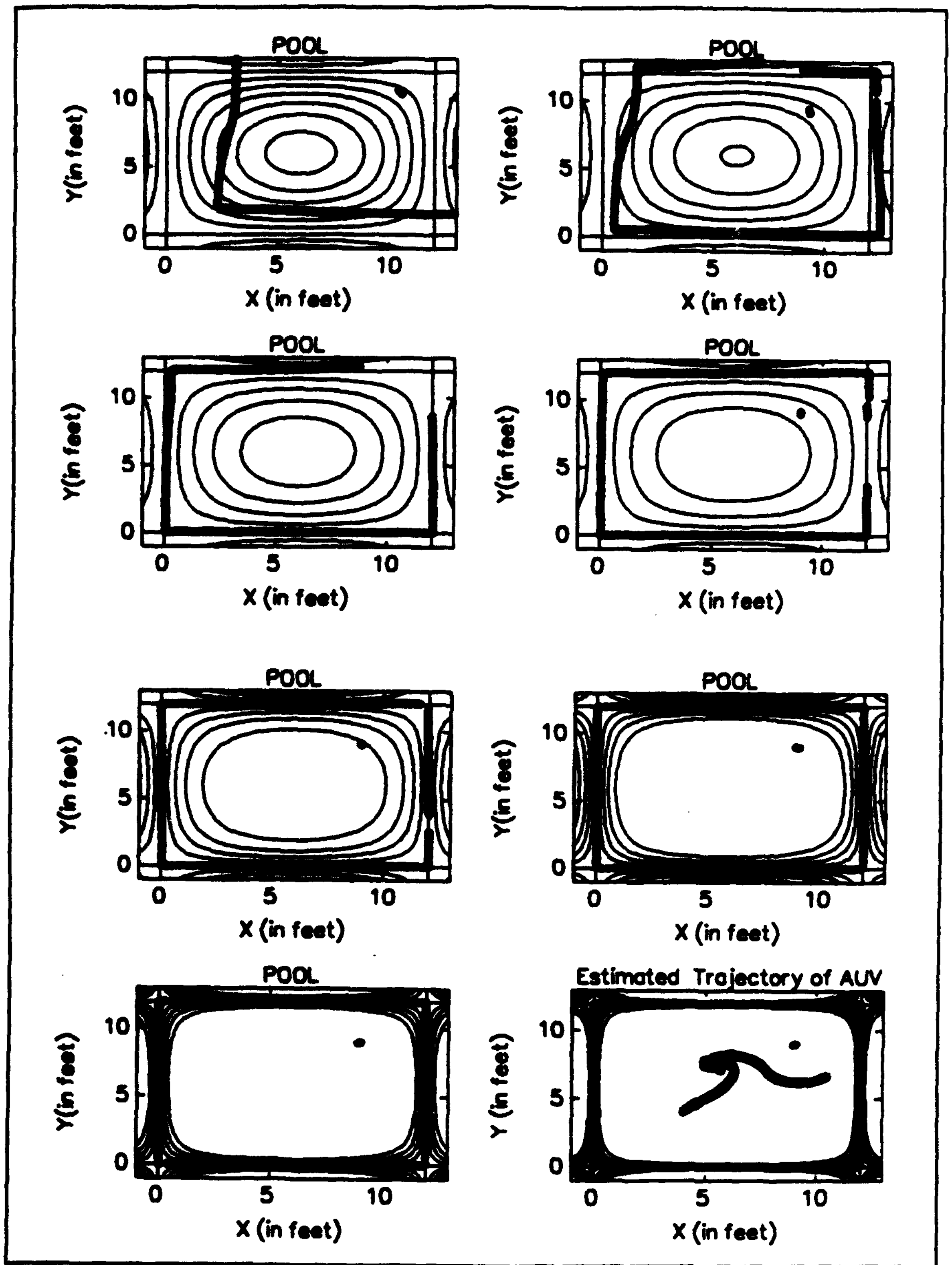

Figure 3.6 Estimation with unknown initial position, the obstacle is not defined by potential function.

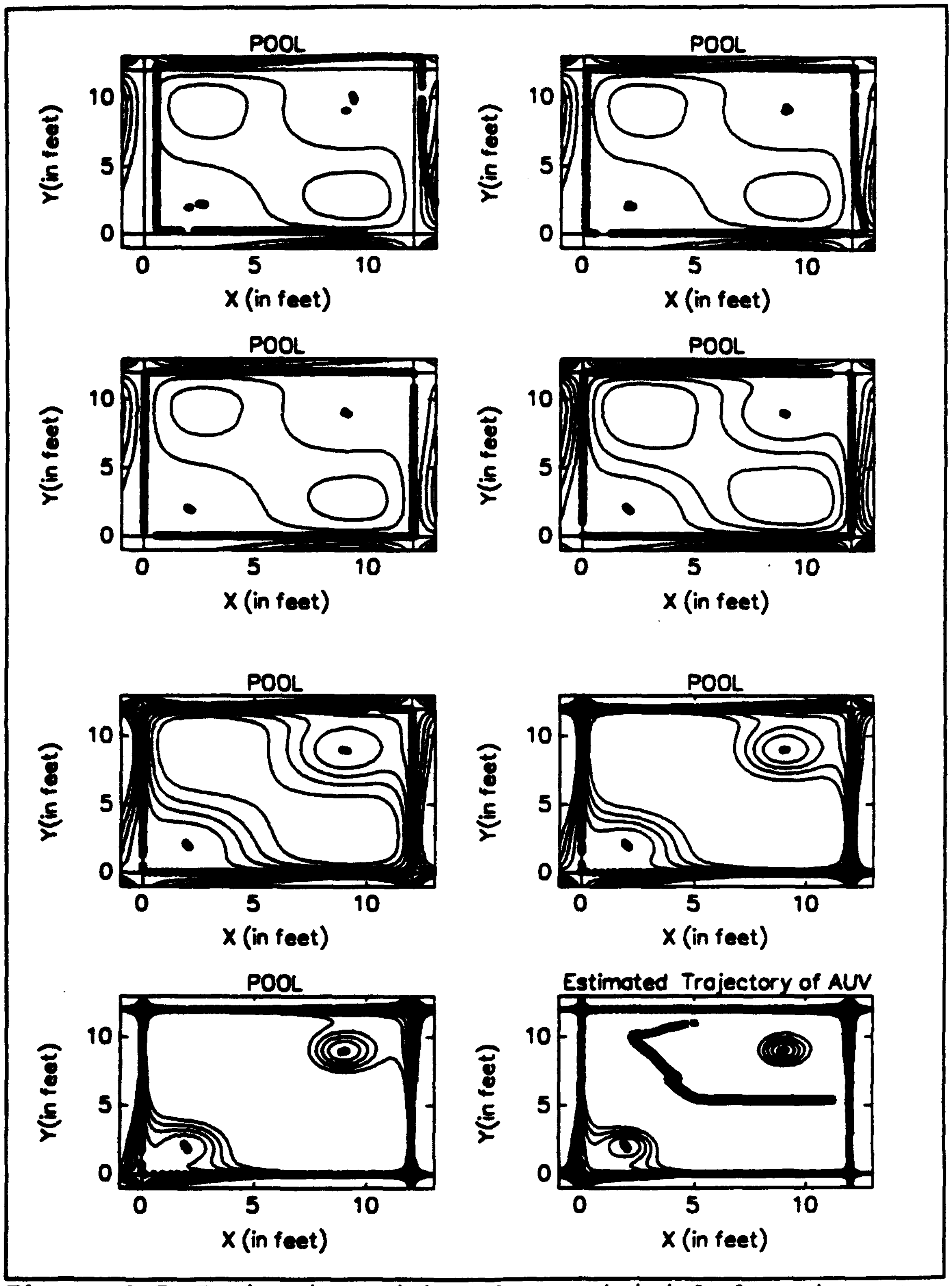

Figure 3.7 Estimation with unknown initial location, two obstacles present.

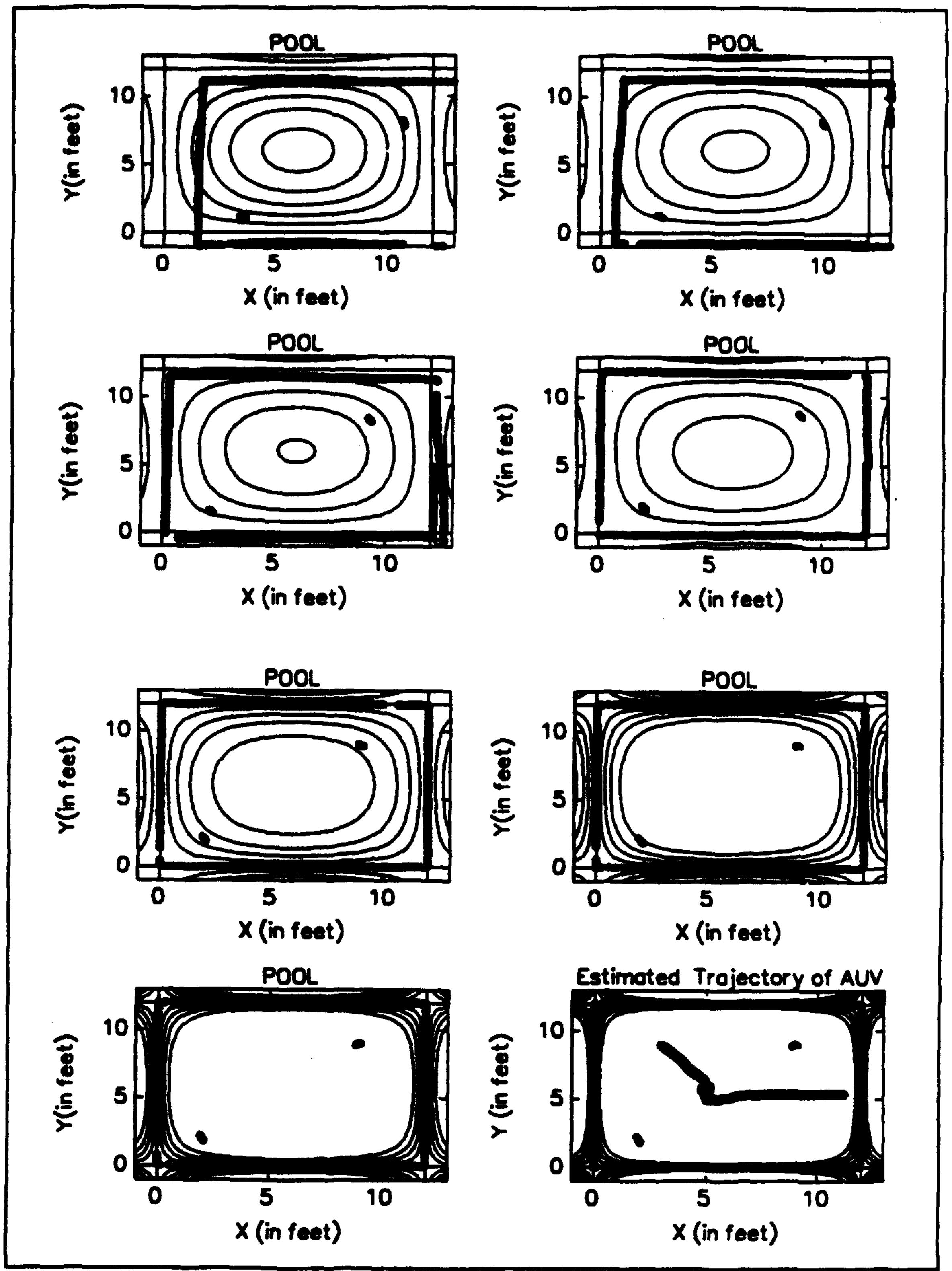

Figure 3.8 Two obstacles present, unknown initial location, the obstacles are not defined by potential function.

D. SIMULATIONS IN AN ENVIRONMENT WITHOUT BORDERS

After testing the algorithm in a pool we proceed further to investigate its behaviour in an environment without borders, such as the open sea. In this case the potential function only defines the obstacles in the area of operation. For the case when one obstacle of cylindrical shape is present, the potential function will be:

$$V(x,y) = (x-a)^2 + (y-b)^2 - r^2. \qquad (3.15)$$

It returns a value of zero if the sonar range return is from the charted obstacles. A 3D plot of the potential function, for the case of one obstacle is shown in Figure (3.9).

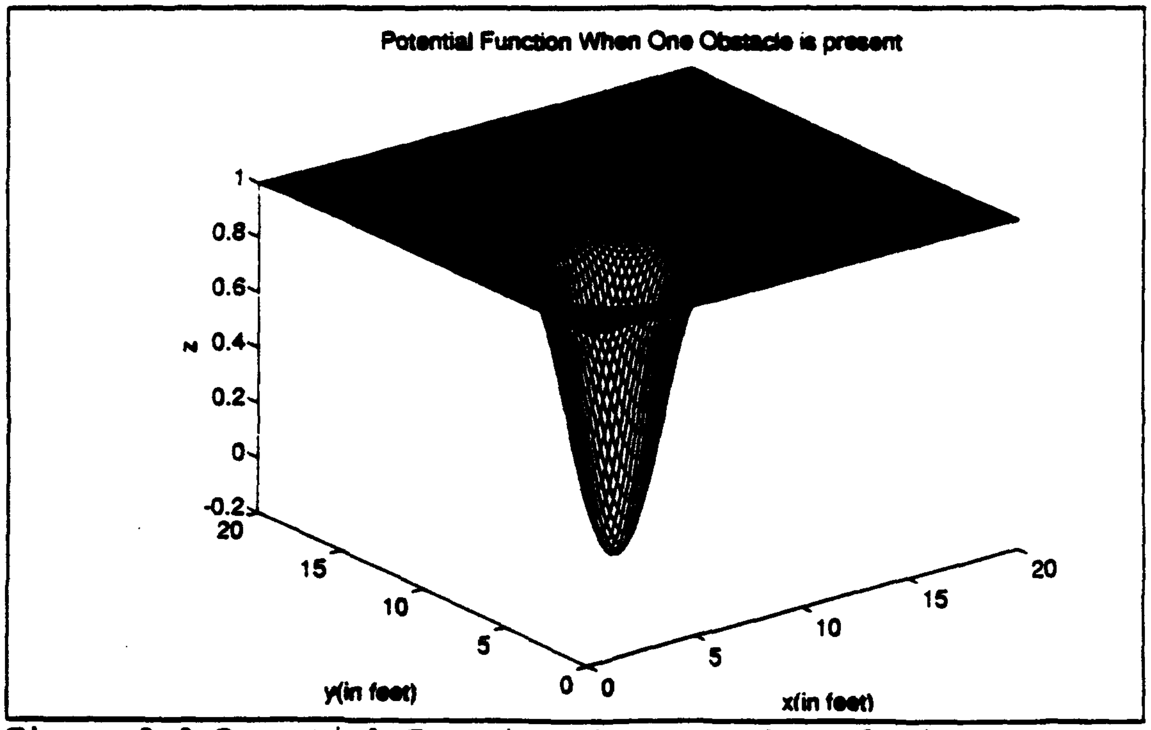

Figure 3.9 Potential Function when one obstacle is present.

In generating data for this simulation a high value of sonar range is given unless the sonar return is from an obstacle. We also processed the data from seven successive sonar scans using a time varying λ parameter. Figure (3.10) shows the estimation when there is an obstacle of cylindrical shape at point (5,5) in the environment. We also assume to have exact information of the initial location of the vehicle. As we notice, the actual and estimated trajectories coincide. In Figure 3.11 we show the position estimation when there is an error in the initial position of the vehicle. In this case the estimated trajectory eventually converges to the actual trajectory. In the second simulation we included three cylindrical buoys at locations (5,5), (15,15), (18,5) so that the potential function description is:

$$\begin{aligned}
\beta_1(x,y) &= (x-5)^2+(y-5)^2-0.25 \\
\beta_2(x,y) &= (x-15)^2+(y-15)^2-0.25 \\
\beta_3(x,y) &= (x-18)^2+(y-5)^2-0.25 \\
V(x,y) &= F_\lambda(\beta_1(x,y)\beta_2(x,y)\beta_3(x,y)) .
\end{aligned} \tag{3.16}$$

A 3D plot of the potential function is shown in Figure (3.12). The result of the estimation is shown in Figure(3.13). In Figures (3.14) and (3.15) we included two extra obstacles at points (3,10) and (8,20) which are not included in the map of the environment. In both cases the algorithm determines the obstacles that are not on the map and marks them on the subsequent plots.

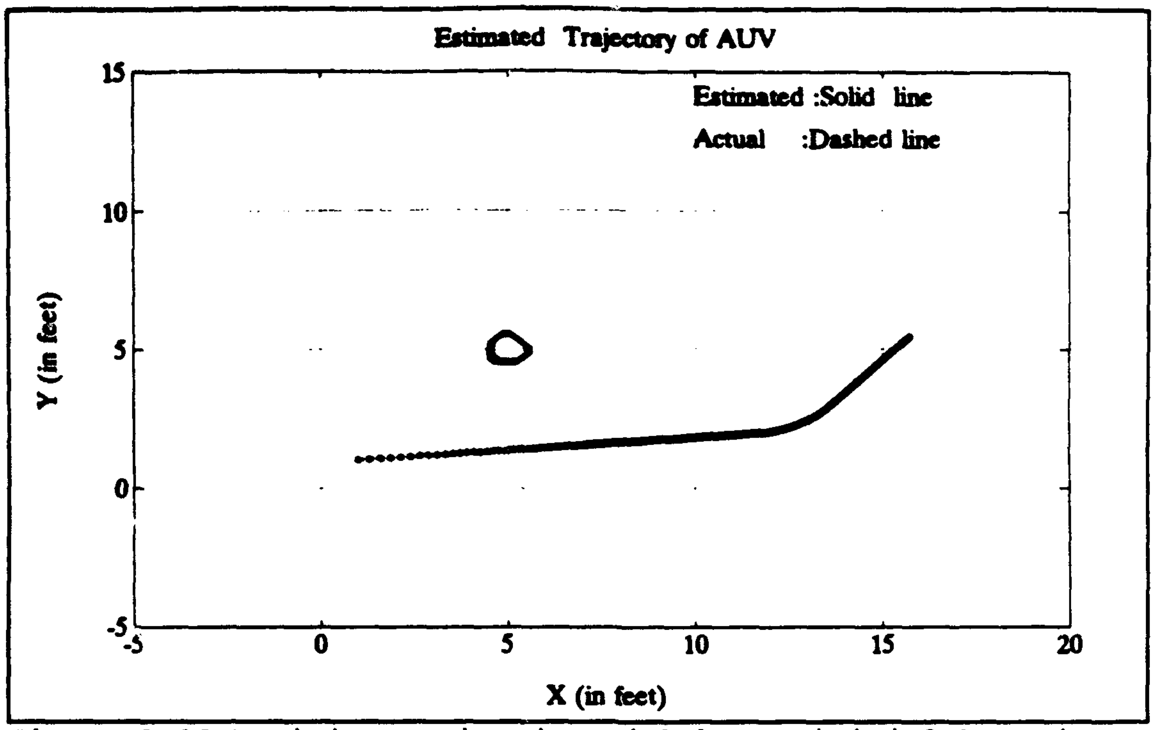

Figure 3.10 Position estimation with known initial locations, one obstacle present.

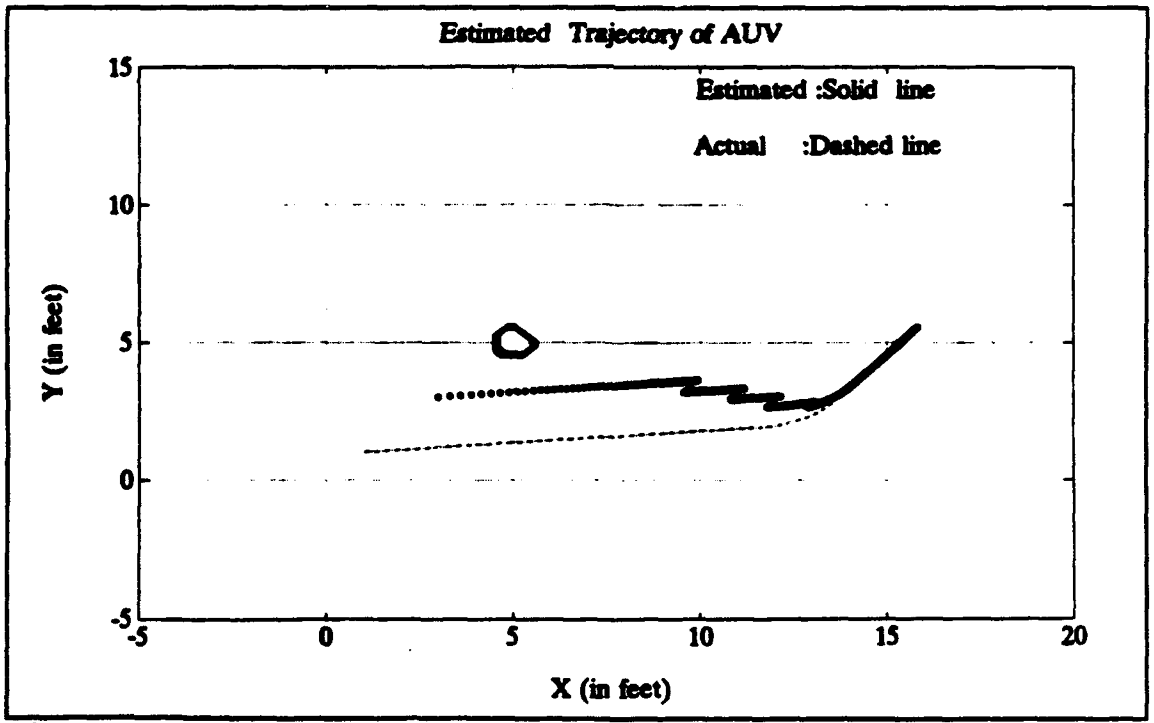

Figure 3.11 Position estimation with unknown initial position, one obstacle present.

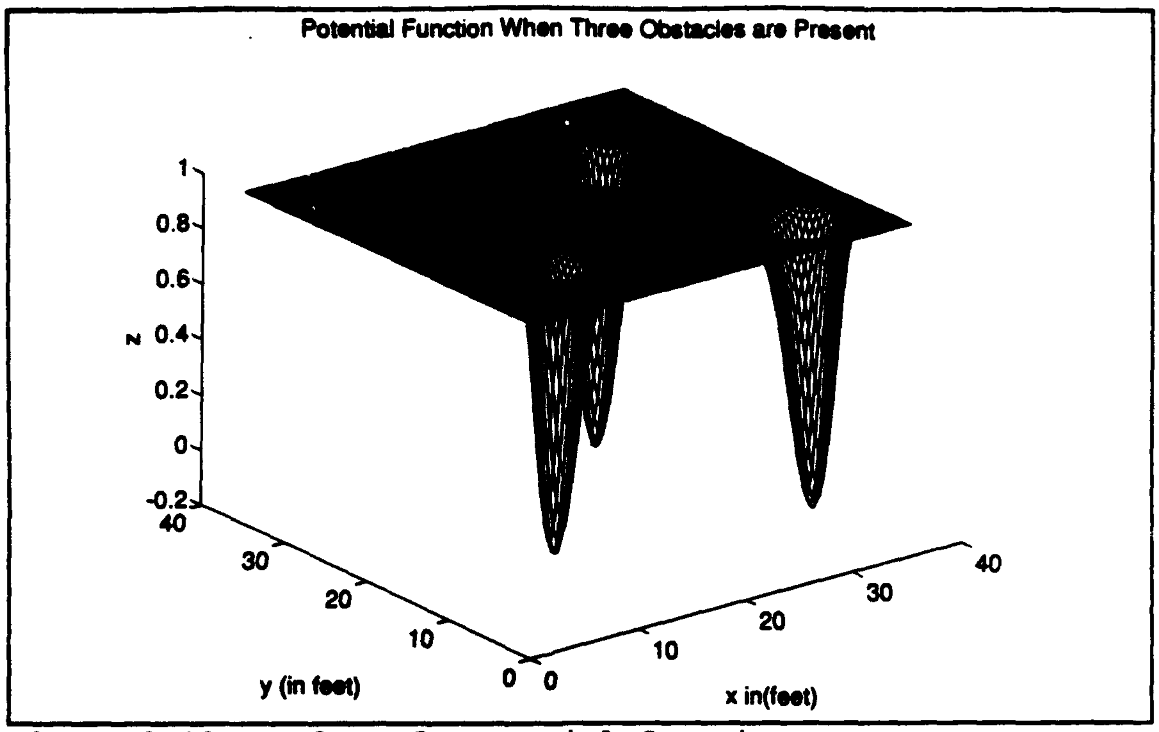

Figure 3.12 3D plot of potential function.

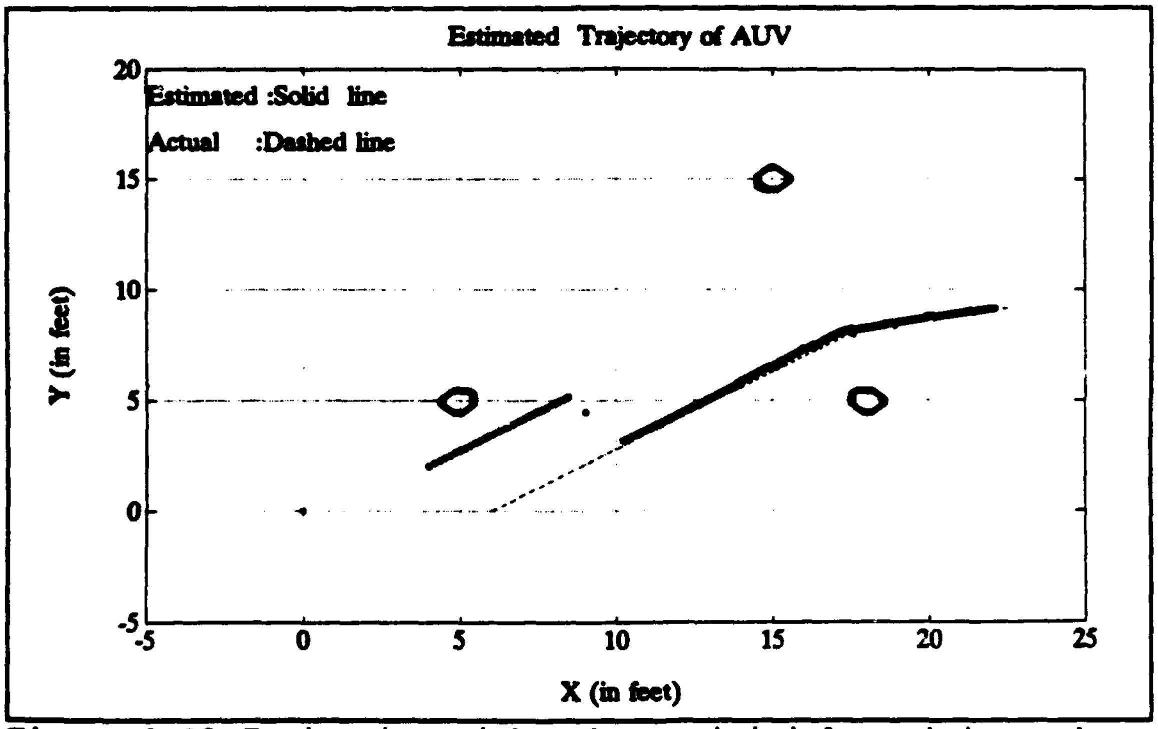

Figure 3.13 Estimation with unknown initial position, three obstacles present.

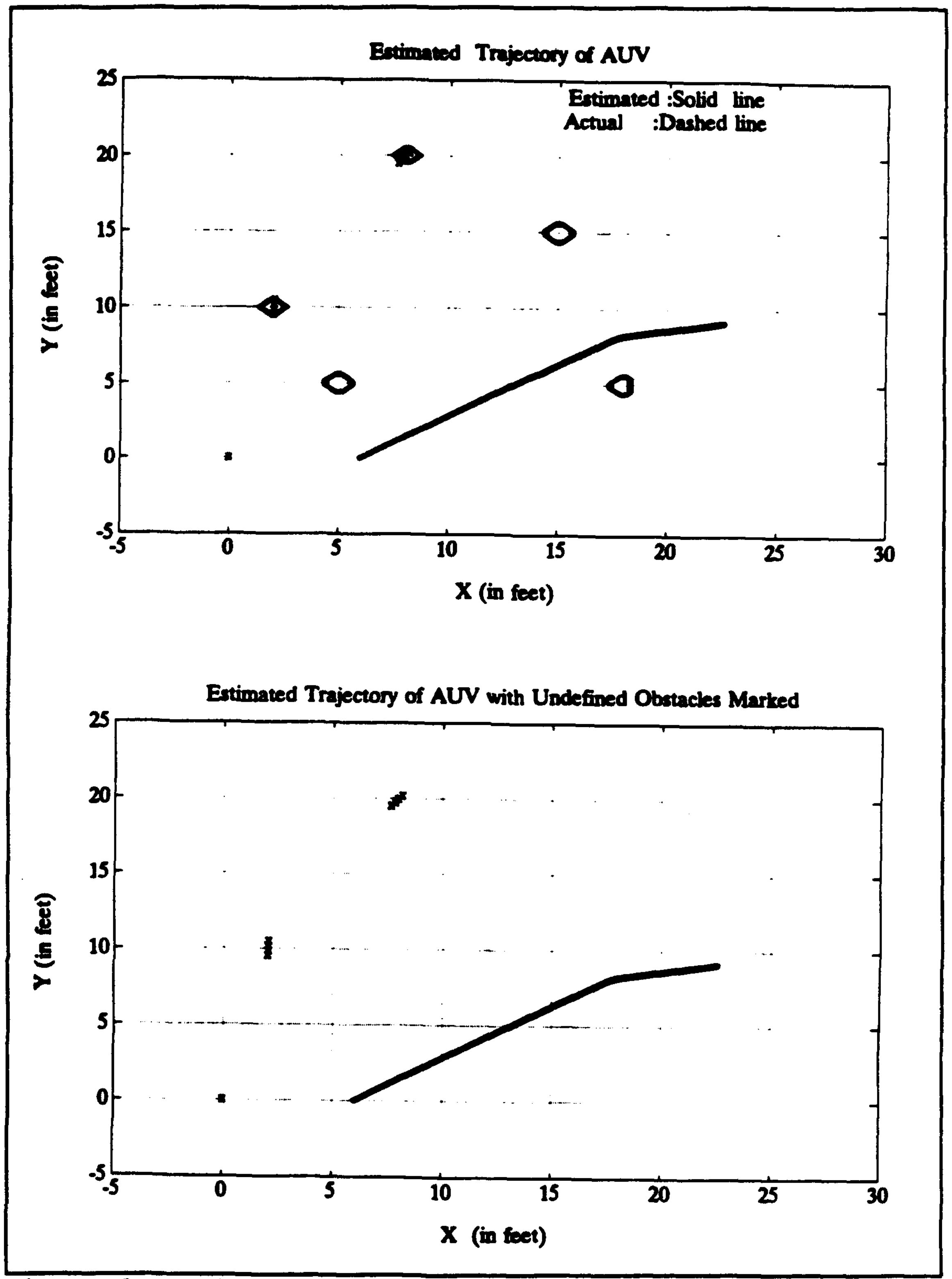

Figure 3.14 Obstacles that are not on the map are marked, known initial position.

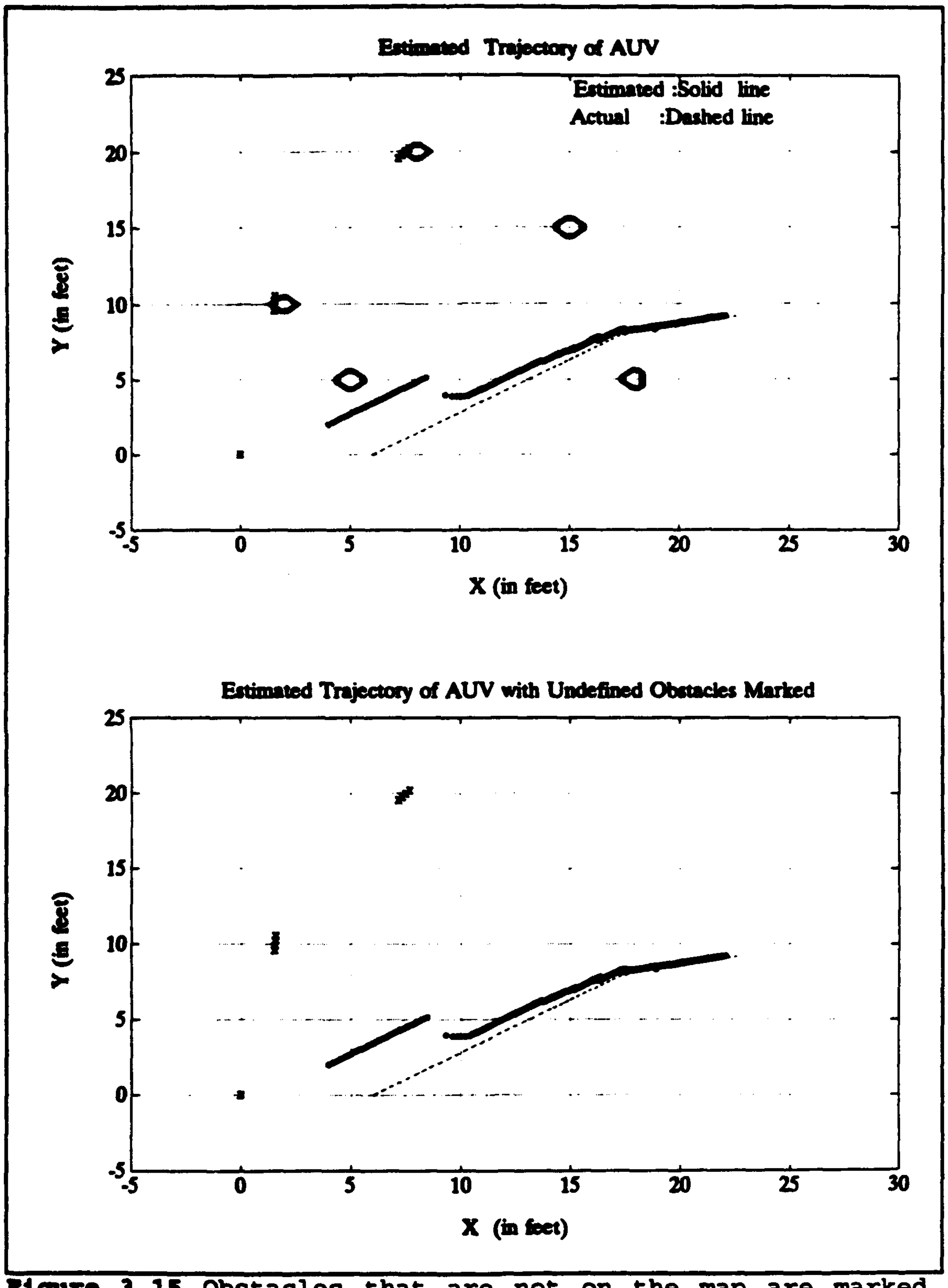

Figure 3.15 Obstacles that are not on the map are marked, unknown initial position.

E. THE EFFECTS OF CURRENTS

Currents present in the area of operation is one of the main contributions to the uncertainty in the position of the vehicle. The estimation algorithm should be modified to accommodate the effects of currents in the navigation algorithm. The inclusion of currents leads to the addition of two extra states in the system model. We further assume the currents to be constant or slowly varying. The two extra states are C_x and C_y which are associated to the components of the currents in the x and y directions respectively, such that:

$$\begin{aligned} \dot{C}_x &= 0 \\ \dot{C}_y &= 0 . \end{aligned} \qquad (3.17)$$

The system states becomes $\mathbf{X} \in \mathbf{R}^7$ and the system model is represented as:

$$\dot{\mathbf{X}} = \begin{bmatrix} 0 & 0 & \cos\theta & 0 & 0 & 1 & 0 \\ 0 & 0 & \sin\theta & 0 & 0 & 0 & 1 \\ 0 & 0 & -a_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -b_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{X} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ a_2 & 0 \\ 0 & 0 \\ 0 & b_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u1 \\ u2 \end{bmatrix} + \mathbf{W}. \qquad (3.18)$$

In Figure (3.16) the effects of currents which cause a significant amount of offset in the final position of the vehicle is shown. Figure (3.17) and Figure (3.18) show the estimation for known and unknown values of the initial position assuming the currents are known. We further tested the algorithm for the case when we do not have any information about the currents in the area of operation, which might generally be the case. Figure (3.19) gives the results of r [illegible]ion estimation, which shows a reliable estimate in the presence of unknown currents.

In Figures (3.20) and (3.21) we repeat the experiment for the case when there are two obstacles in the area. For both cases the navigation algorithm is able to recover the actual trajectory.

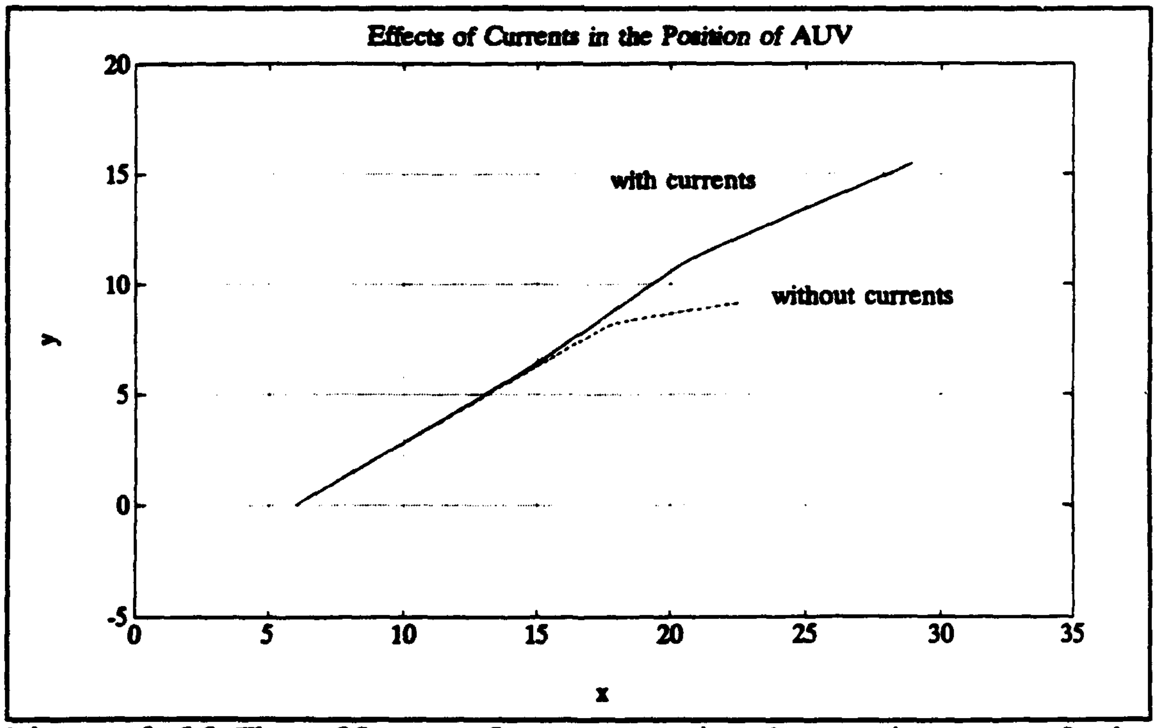

Figure 3.16 The effects of currents in the trajectory of the vehicle.

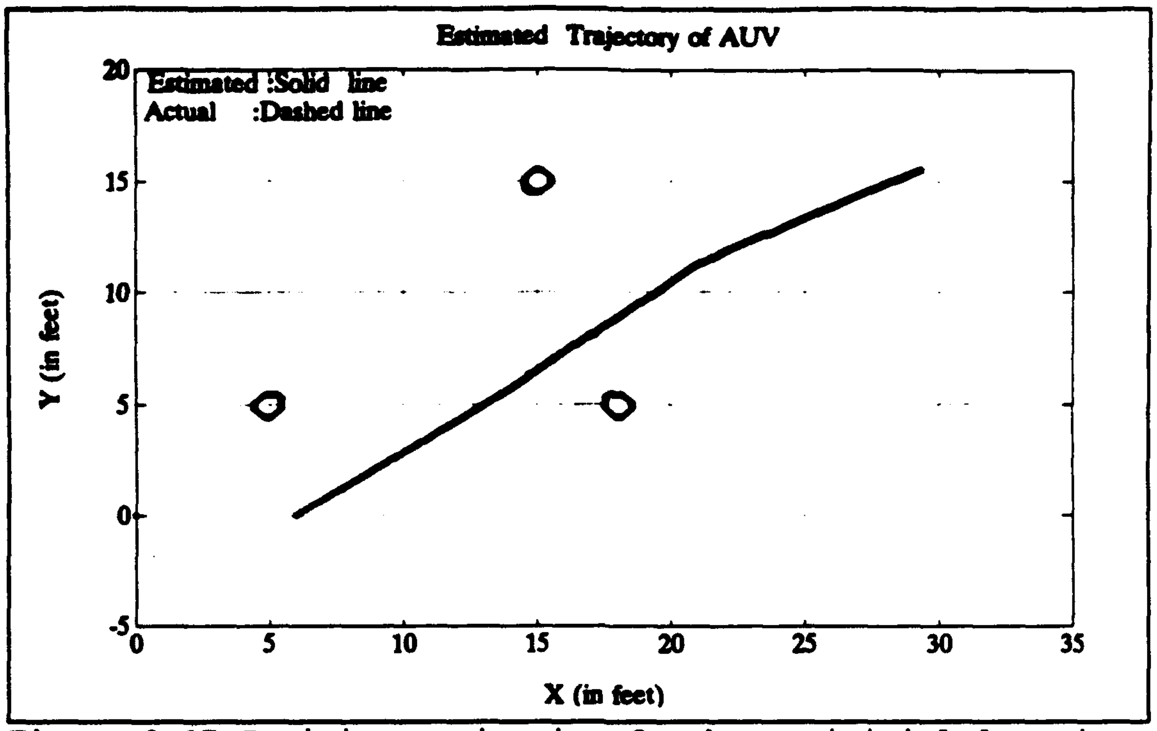

Figure 3.17 Position estimation for known initial location, three obstacles present.

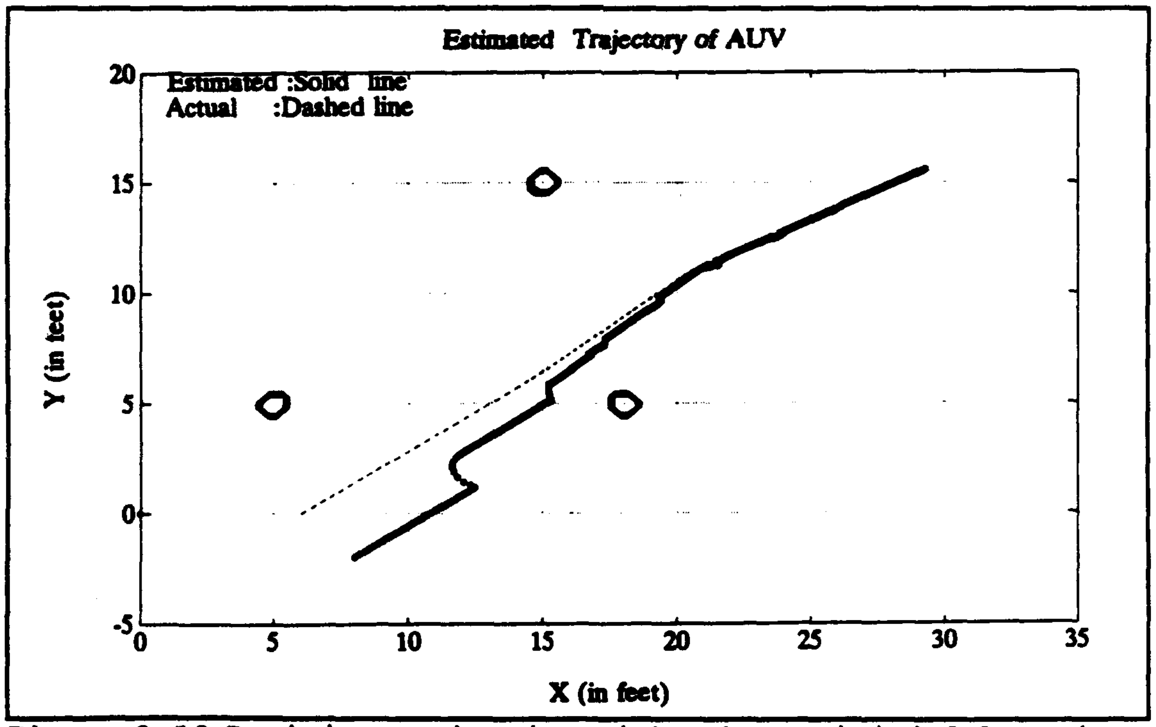

Figure 3.18 Position estimation with unknown initial location, three obstacles present.

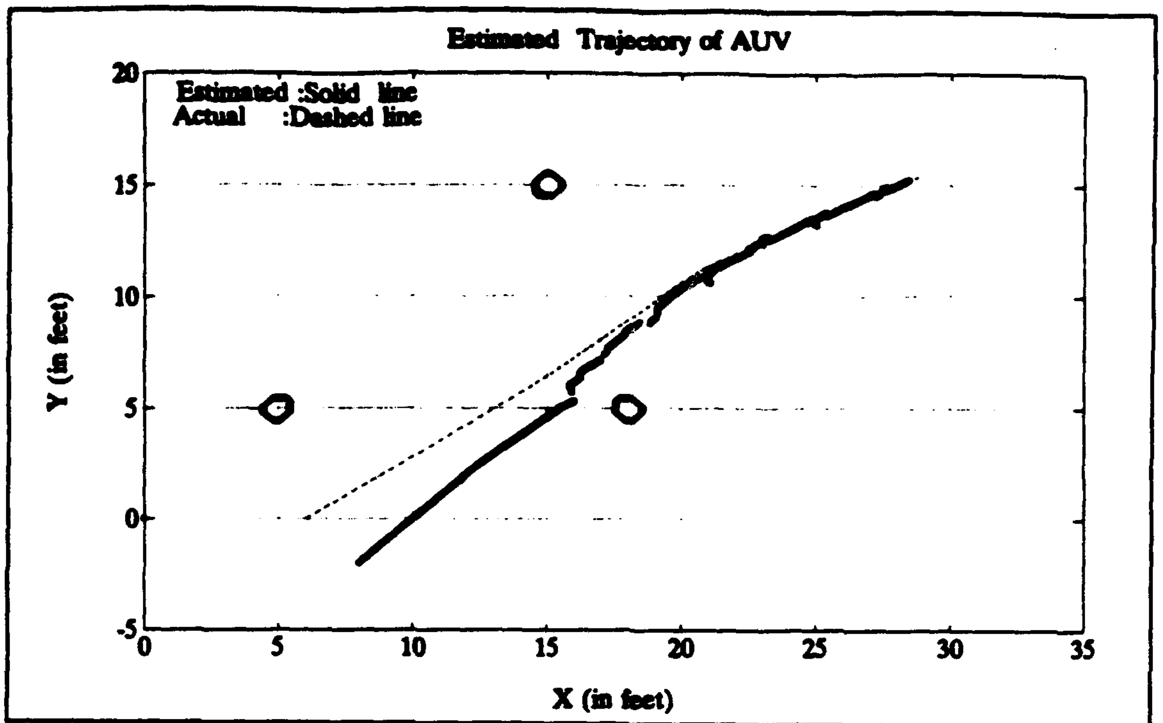

Figure 3.19 Position estimation with unknown initial position and current, three obstacles present.

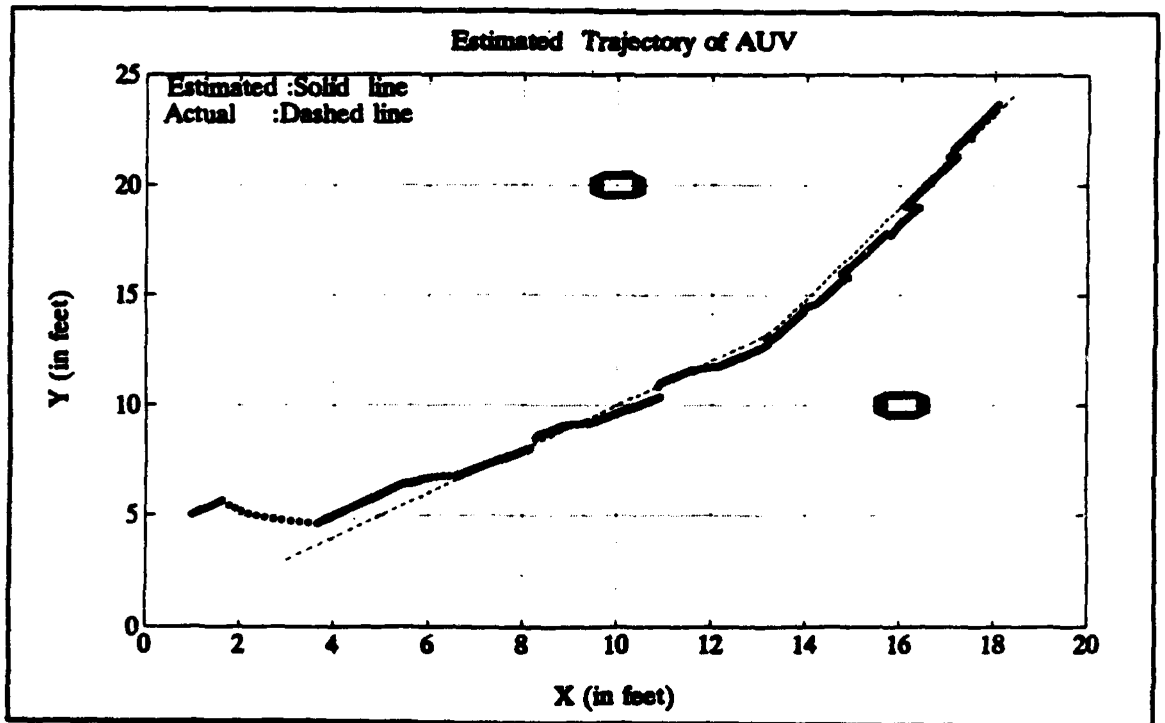

Figure 3.20 Position estimation with unknown initial position and currents, two obstacles present.

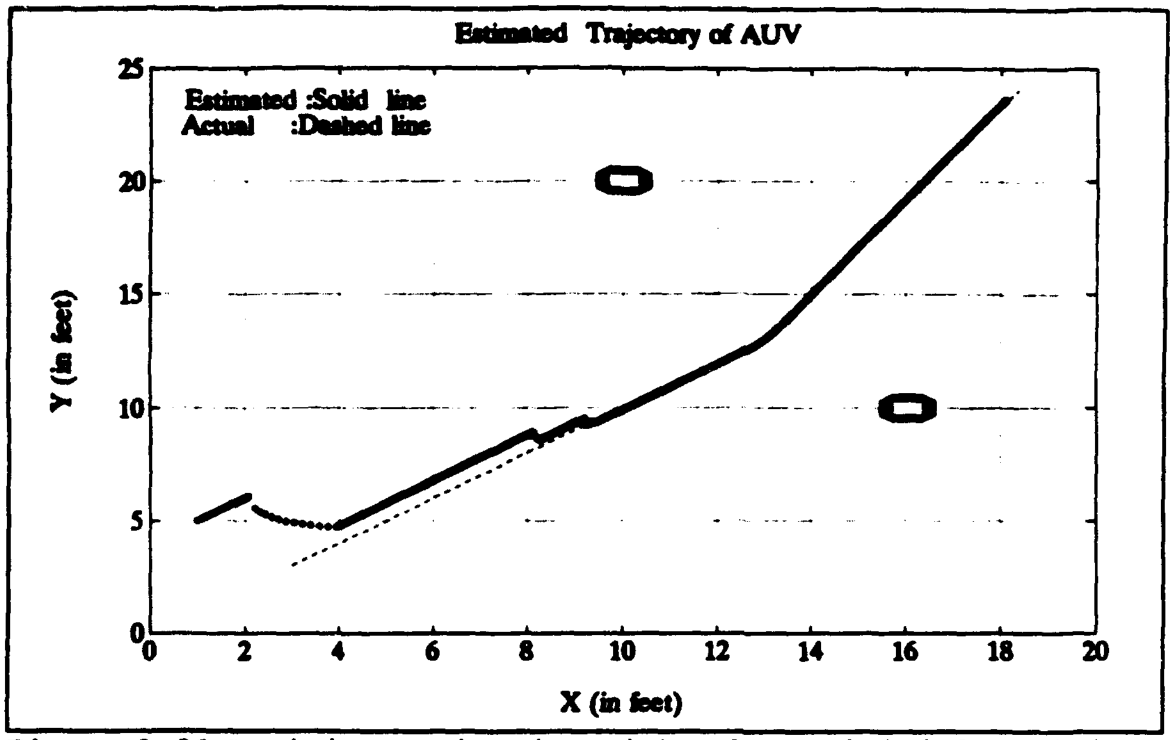

Figure 3.21 Position estimation with unknown initial position, two obstacles present.

For the simulations in this Chapter we have applied Least Squares (LS) parameter estimation techniques to estimate dynamic parameters a_1, a_2, b_1, b_2 associated with Equation (2.44).

The results of the parameter estimation are given in Table (3.1). The actual values are used in data generation for the computer simulations. The table shows the mean of the estimated parameters for six different simulations. The percentage error made in the estimates are less than one percent which is within acceptable limits.

Table 3.1 ESTIMATED DYNAMIC PARAMETERS BY LEAST SQUARES.

Dynamic Parameter	Actual Value	Estimated Value	Percentage Error
a_1	1.0000	0.9977	0.23 %
a_2	0.0010	0.0010	0.0 %
b_1	1.0000	1.0002	0.02 %
b_2	1.0000	1.0001	0.01 %

F. STUDIES WITH ACTUAL DATA

In this section we apply the navigation algorithm using actual data. The data is collected on a water tank of size 6x6 meter. The data consists of a set of sonar returns including range ρ and bearing α. The sonar measurements are taken by a Tritech 725 high resolution sonar onboard the NPS AUVII. Five successive sonar scans are processed by the Kalman filter. Each sonar scan consists of 200 range measurements corresponding to a sector of 1.8 degrees per sonar return. The typical sonar scan of the pool is shown in Figure (3.22). Due to gain problems in the sonar, the range measurements are excessively corrupted by noise. We see these effects in the sonar scan of Figure (3.22), even though there are no objects inside the watertank we have sonar readings inside the tank as well as the outside of it. In collecting data for the experiment, no RPM or rudder command was given to the vehicle and constant speed and heading was used. Therefore, we used a simpler version of the system model. In this case our state vector is $\mathbf{X} \in \mathbf{R}^4$, given by x and y position, velocity and heading. This results in a state space form of :

$$\dot{\mathbf{X}}=\begin{bmatrix} 0 & 0 & \cos\theta & 0 \\ 0 & 0 & \sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\mathbf{X}+\mathbf{W}, \qquad (3,19)$$

where $\mathbf{W}$ is zero mean gaussian white noise. In this case we do not have any obstacles in the tank so the potential function will only define the borders of the pool such that:

$$V(x,y)=F_\lambda(x(x-6)y(y-6)). \qquad (3,20)$$

The range and bearing values from each successive sonar return are fed to the Kalman Filter and the parameter λ is reduced by 50 percent at each complete scan. The position estimation for known initial location is given in Figure (3.23). In the subsequent plots, estimated borders of the pool for each successive sonar scan are plotted along with the contours of the potential function. Also the estimated trajectory of the vehicle is shown. The readings inside the pool are due to the gain problems associated with the sonar as mentioned before. In Figure (3.24) the position estimation with unknown initial condition is shown, where we have the center of the tank, namely the point (3,3) as an initial estimate. The estimation converges to the actual values.

The fixed interval smoothing, as explained in Chapter II, is an off-line procedure that uses the state estimates and associated error covariances to smooth the estimates produced by the Kalman Filter. In order to overcome the transient problems in the filter and get better estimates about the position of the AUV we applied the smoothing algorithm to the Kalman filter output. In applying the algorithm we stored the state estimates and the error covariances for each scan and fed these values to the smoothing filter. The smoothing filter runs backwards in time and the final state estimate and the error covariance of the Kalman filter become the a priori estimate and the error covariance of the smoothing filter.

In Figures (3.25) and (3.26) results of the estimation with a smoothing filter are shown. The smoothed trajectory is given in the last plot. It is easily seen that fixed interval smoothing provides better state estimates and does not suffer the transient problems associated with the Kalman filter. In Figure (3.27) we applied the Fixed Interval Smoothing algorithm to the computer generated data. The main disadvantage of the smoothing algorithm is having to store a large amount of data, which may cause memory overflow in the computer of the AUV studied in this thesis.

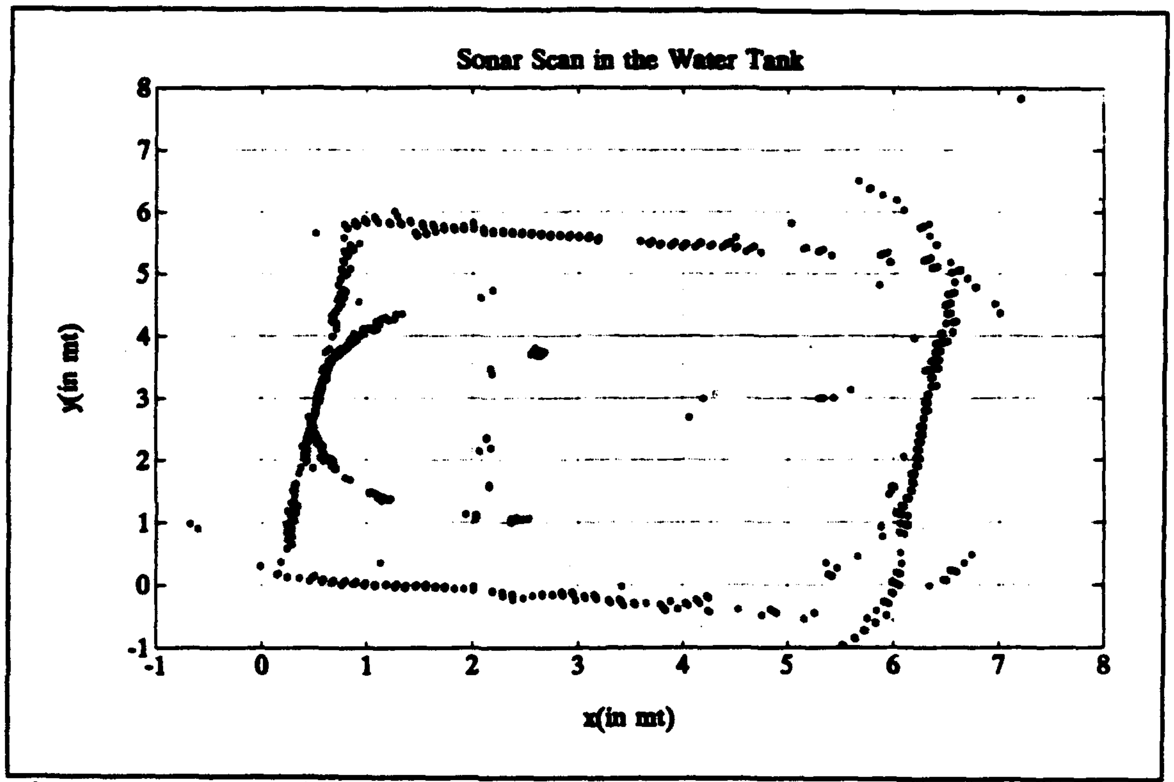

Figure 3.22 Typical sonar scan in the pool.

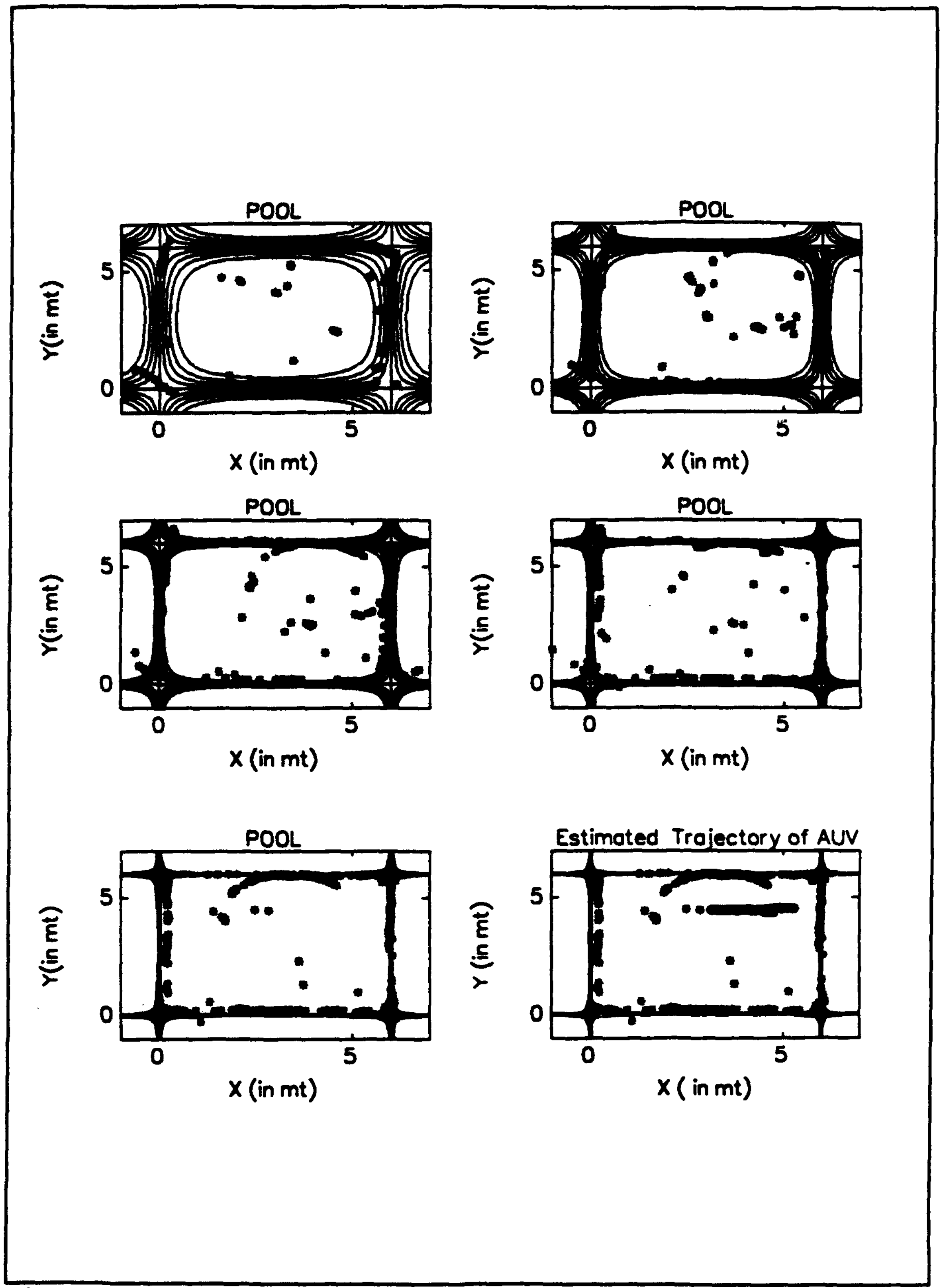

Figure 3.23 Position Estimation with known initial location.

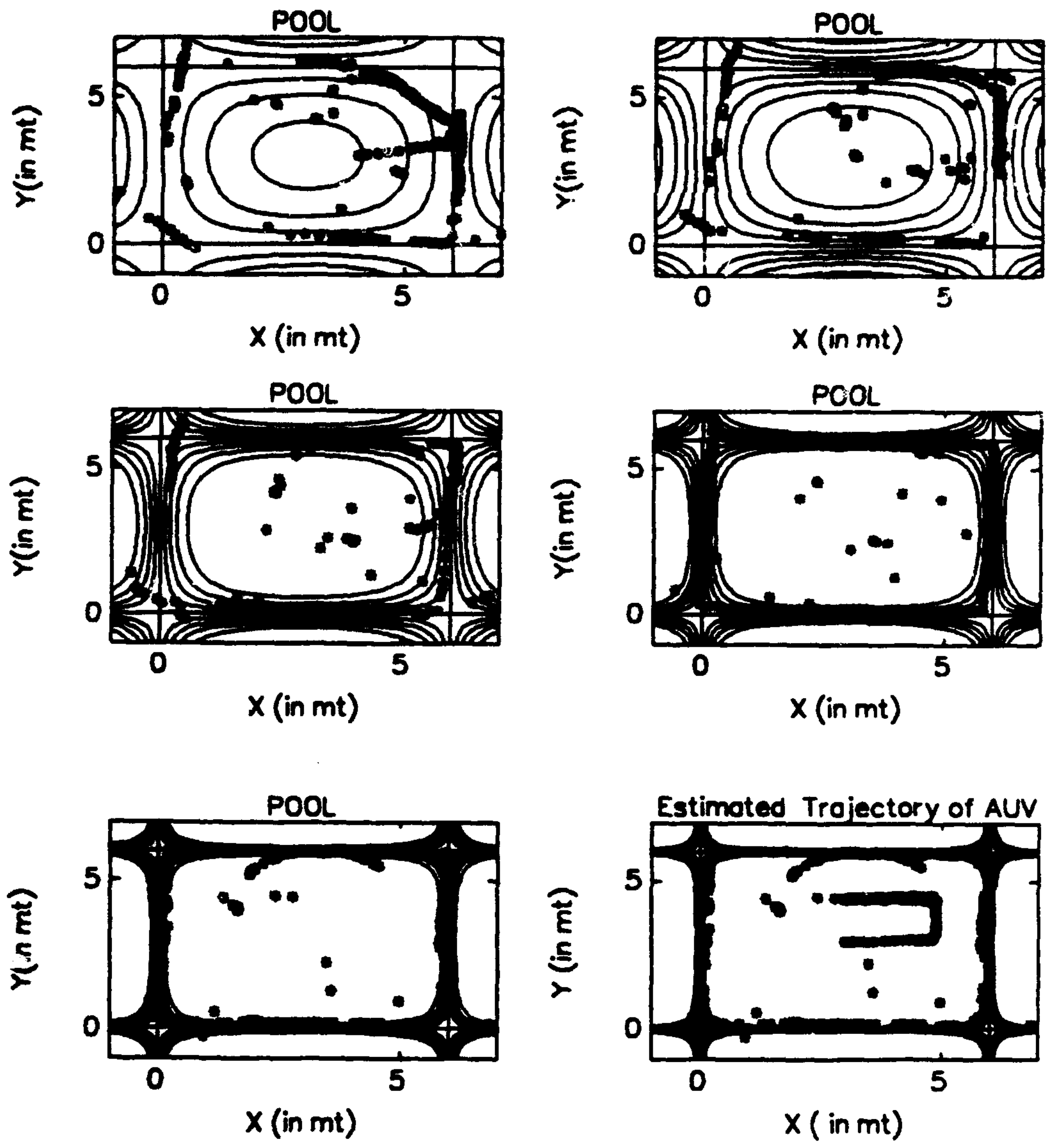

Figure 3.24 Position estimation with unknown initial location.

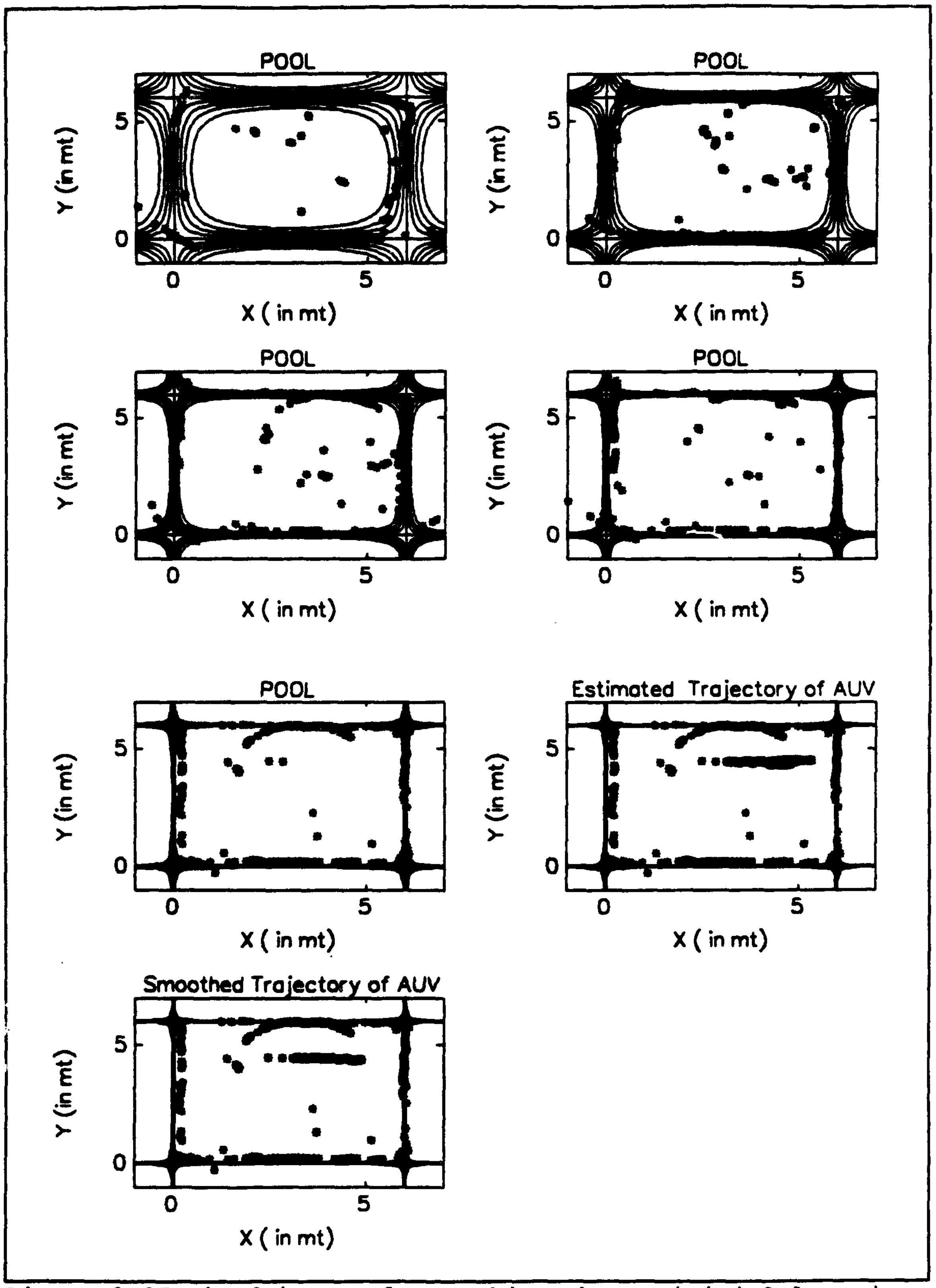

Figure 3.25 Fixed interval smoothing, known initial location.

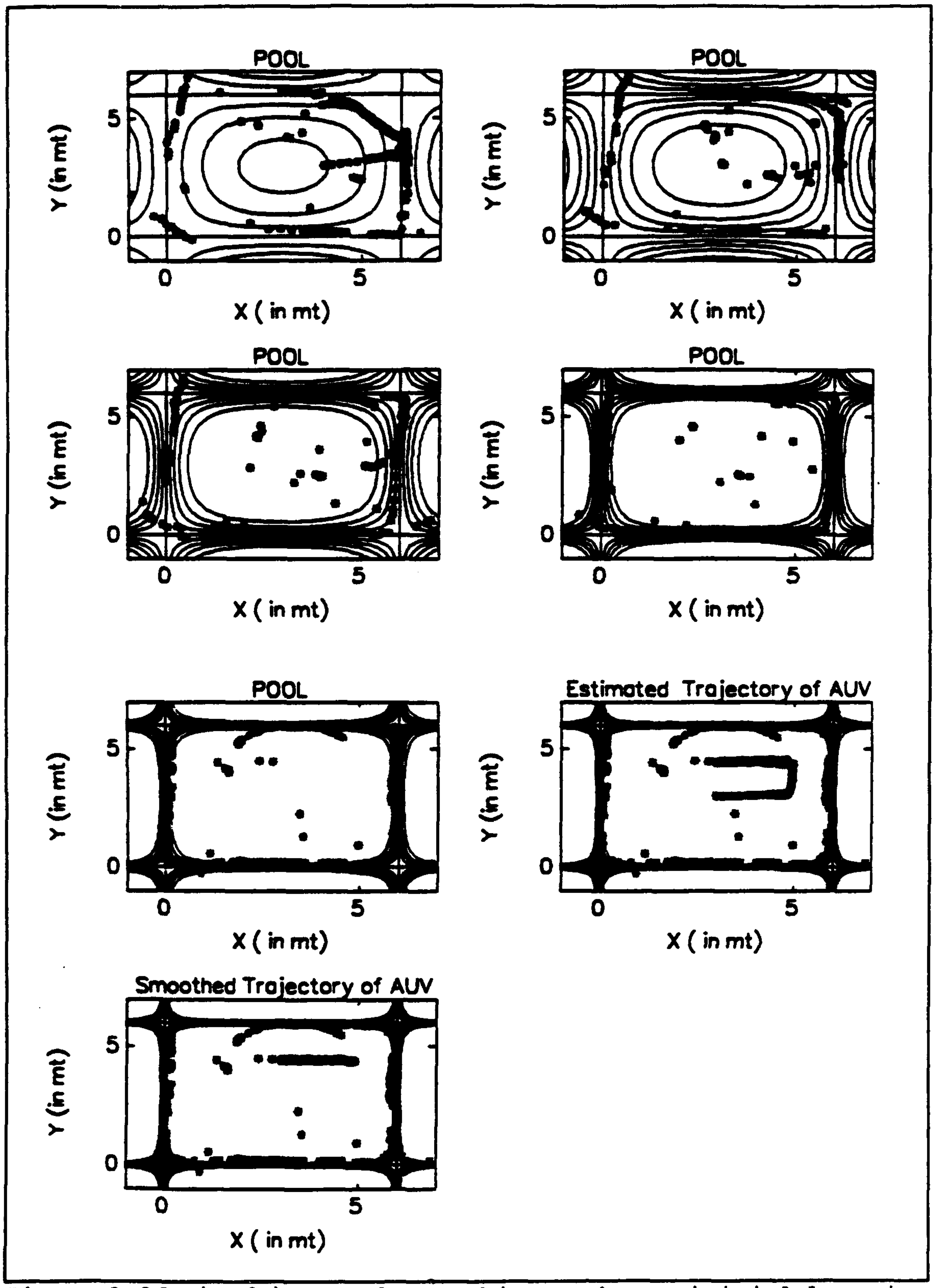

Figure 3.26 Fixed interval smoothing, unknown initial location

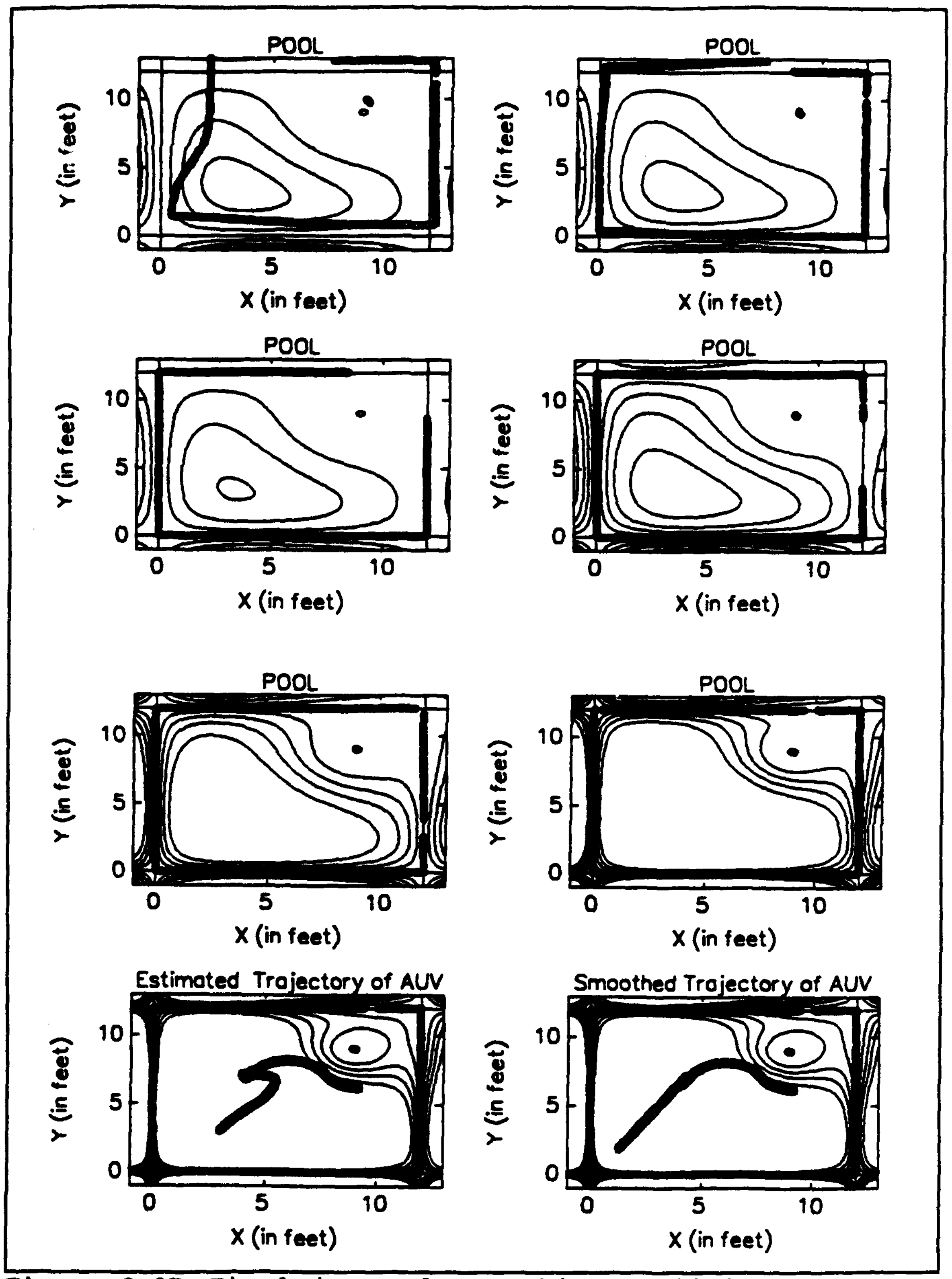

Figure 3.27 Fixed interval smoothing applied to computer generated data.

IV. NEURAL NETWORK MODELS FOR POTENTIAL FUNCTIONS

A. GENERAL

A neural network consists of many simple elements operating in parallel. It can be trained to perform complex tasks involving system identification and classification, control systems, vision and pattern recognition [Ref. 8].

Neural networks are composed of neurons. Each neuron in the net is made of weighted inputs, a bias and a transfer function. The input to a neural network is multiplied by its weight and summed by the bias b. The role of the bias is to shift the argument of the transfer function by a corresponding amount. The model for a neuron is given in Figure (4.1).

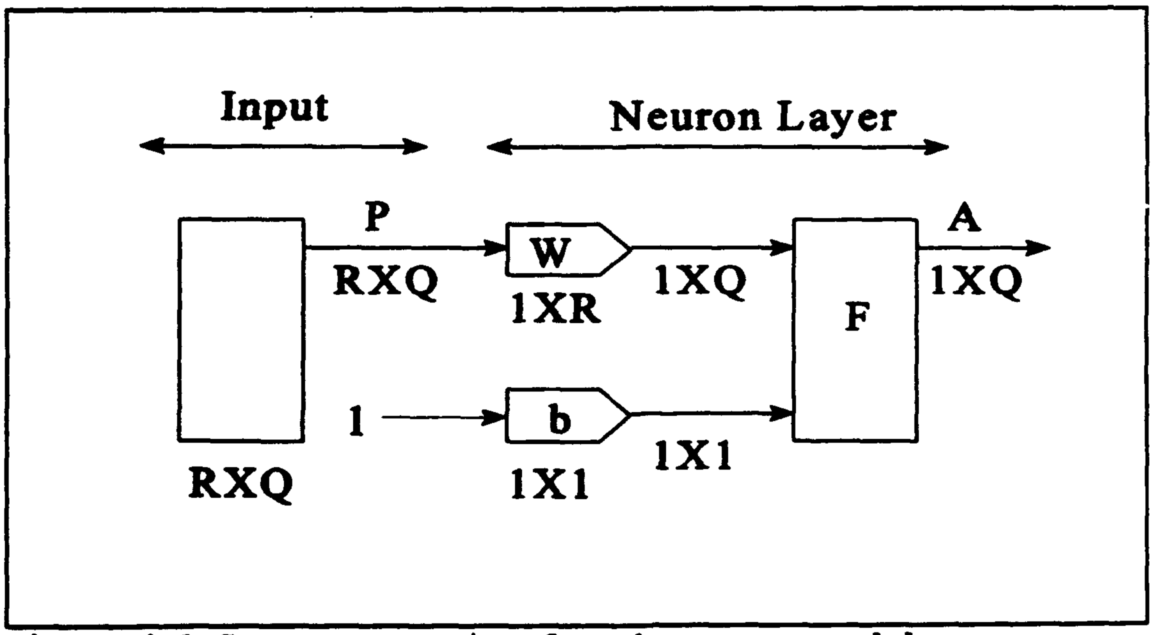

Figure 4.1 Compact notation for the neuron model.

In this representation R represents the number of inputs to the neuron and Q is the number of input vectors which is called *batch*. Two or more neurons can be combined in a single layer and a neural network may contain one or more such layers [Ref. 8].

When a neural network consists of multiple layers of neurons, the layer whose output is the overall network output is called output layer and other layers are called hidden layers. An example of a two layer neural network is given in Figure (4.2). In the hidden layer there are S1 neurons and in the output layer there are S2 neurons.

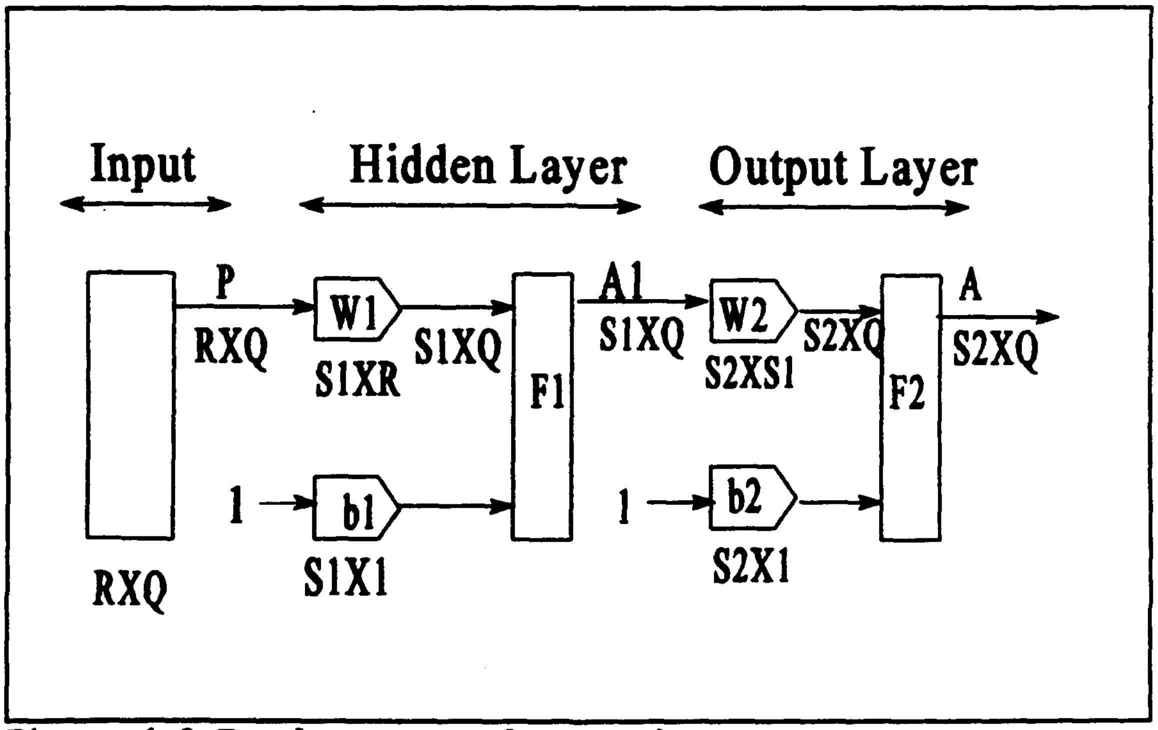

Figure 4.2 Two layer neural network.

The output at the hidden layer is given by:

$$A1 = F1(W1*P+B1)\,. \tag{4.1}$$

The overall network output A is given by:

$$\begin{gathered} A=F2(W2*A1+B2) \\ A=F2(W2*F1(W1*P+B1)+B2)\,, \end{gathered} \tag{4.2}$$

where F2 is the transfer function of the output layer, W2 and B2 are the weights and biases for the output layer respectively.

B. BACKPROPAGATION NEURAL NETWORKS

The backpropagation algorithm is designed to train multi layered neural networks with different transfer functions to perform function approximation and pattern recognition [Ref. 8]. It can be shown that a backpropagation neural network with at least one sigmoid layer is capable of approximating any smooth function.

The algorithm adjusts weights and biases so as to minimize the sum squared error between the network's output and the desired signal. This procedure is done in the direction of the steepest descent with respect to the error which is called *gradient descent* procedure.

Backpropagation neural networks give reasonable answers when they are fed with inputs they have never seen. This generalization property gives the convenience of training the network with fewer input/output pairs instead of the whole input/output data [Ref. 8].

In the backpropagation algorithm the parameters W are updated as:

$$\hat{W}_{k+1} = \hat{W}_k - \mu \frac{\partial J}{\partial W}\Big|_{W=\hat{W}_k} \tag{4.3}$$

where J is the square of the error between network's output and desired signal, and μ is the learning rate. If the learning rate is too high it causes unstable learning, in a sense that the weights will diverge. If it is too small it requires very long training times.

Another complication with backpropagation is the problem of local minima. This is caused by the fact that the error function is non quadratic in the weights. It is possible for the estimated parameters to become trapped in one of the local minima. This problem can be addressed by adding a momentum term. Momentum makes the neural network insensitive to the small variations in the error surface. Therefore it prevents the neural network's solution to converge to a local minimum which causes higher steady state error.

C. MODELING OF POTENTIAL FUNCTIONS BY NEURAL NETWORKS

In this thesis we use backpropagation neural networks in the context of function approximation. The main idea behind it is to get approximate values of the potential function and its derivatives for use in the Kalman filter. As stated in Chapter III, the nonlinear measurement equation is the potential function. The linearized H matrix is therefore made of the derivatives of the potential function with respect to the states:

$$H=\left[\frac{\partial V}{\partial x},\ \frac{\partial V}{\partial y},\ 0,\ \frac{\partial V}{\partial \theta},\ 0\right]. \tag{4.4}$$

When the system detects a reflecting surface at point (x,y) in the plane, the network is trained with a "zero" output in accordance with the definition of the potential function as in Chapter III.

The inputs to the neural network are the coordinates of the tip of the sonar beam which is a two element input vector. The potential function with its derivatives is a three element output vector which we use to train the network. These input and output pairs are nonlinearly related. We apply a backpropagation algorithm for our network model. The network architecture for this model is shown in Figure (4.3).

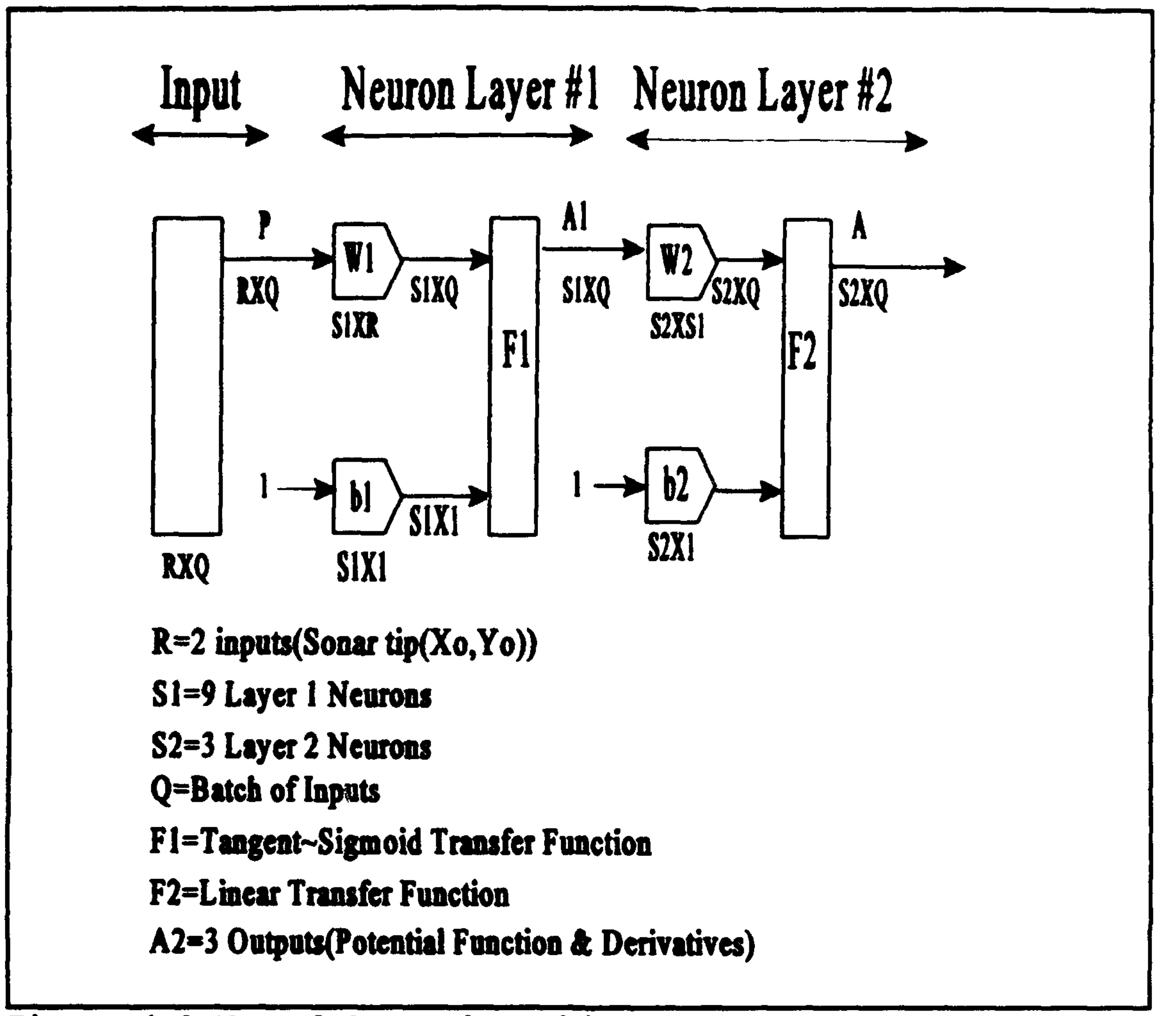

Figure 4.3 Neural Network architecture.

As the input to the network we have the coordinates of the tip of the sonar beam. In layer one, nine neurons with a tangent-sigmoid transfer function are used. In layer two, since our output size is three, only three neurons are used.

The results of training for the potential function are given in the following figures. In Figure (4.4) a two layer plain backpropagation network is used. After 300 epochs the sum squared error reaches a minimum value. Figure (4.5) shows the results of training when backpropagation with added momentum term is used. While the plain backpropagation converges to a local minimum, the momentum makes the network converge to a lower sum squared error.

The training time can also be decreased by using an adaptive learning rate by optimizing the μ based on the error. Figure (4.6) gives the results of training when an adaptive learning rate with momentum is used. The learning process is faster; while it takes 300 epochs to reach a minimum sum squared error with momentum only, the same error level can be reached within 120 epochs by adding an adaptive learning rate. Consequently a two layer backpropagation neural network with adaptive learning rate and momentum is used for our application.

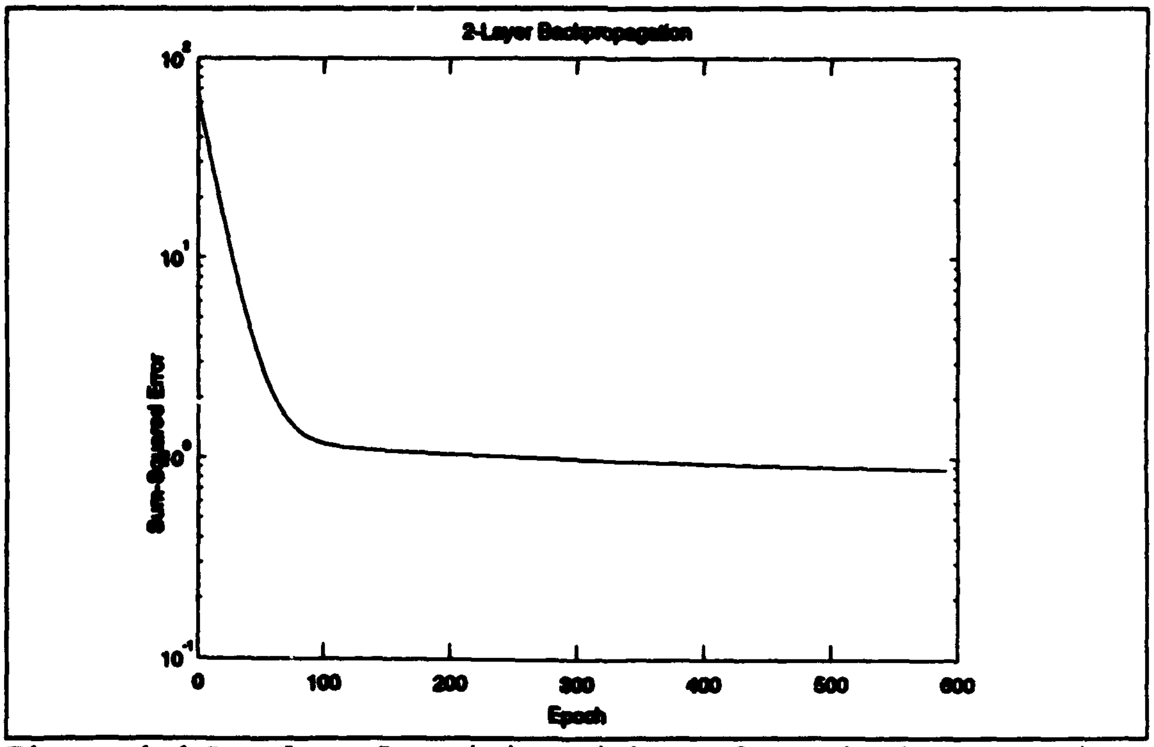

Figure 4.4 Results of training with two layer backpropagation.

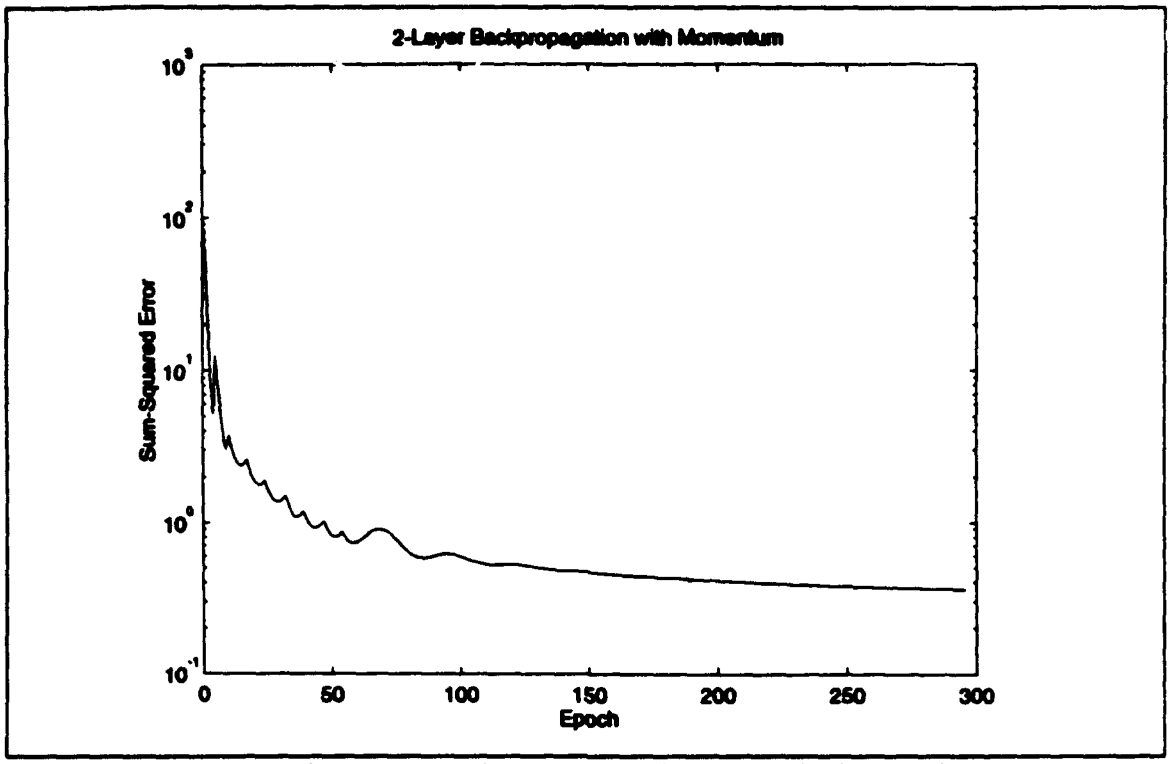

Figure 4.5 Two layer backpropagation training with momentum term.

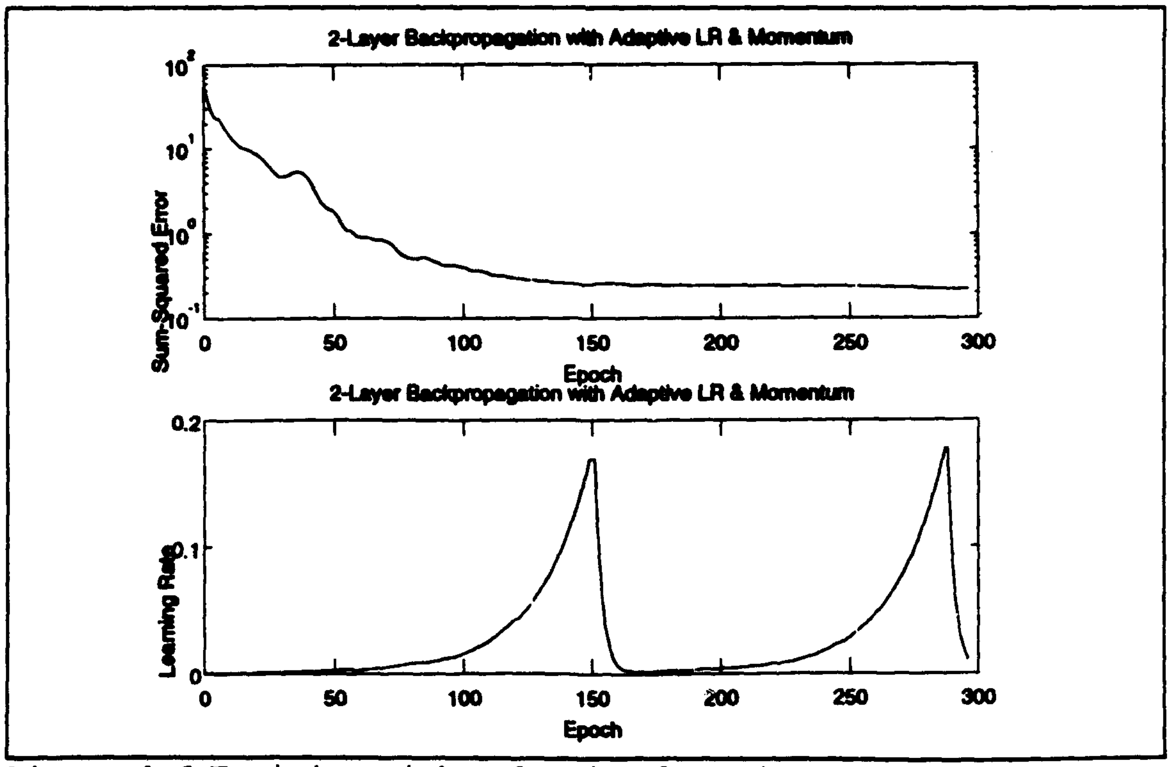

Figure 4.6 Training with adaptive learning rate and momentum.

After training the neural network with input and output data sets as explained before, the values of the weight and bias matrices are stored. After this, the potential function is computed using the neural network. During the estimation, the coordinates of the sonar tip (x_0,y_0), are fed to the neural network. The values of the potential function and its derivatives are taken as the network's output, and used by the Kalman filter. However, the training of the neural network for the case of unknown initial conditions of the vehicle was not successful due to the uncertainties in the problem.

The results of the estimation based on the backpropagation neural network are given in the following figures. In Figure (4.7) there is one obstacle located at point (15,15) and in Figure (4.8) there are two obstacles located at (5,10) and (10,5). In both cases the estimated position follows the actual trajectory of the vehicle. Finally, in Figure (4.9), the algorithm is applied to the actual data collected in the water tank, which exhibits fairly good estimates about the position of the vehicle.

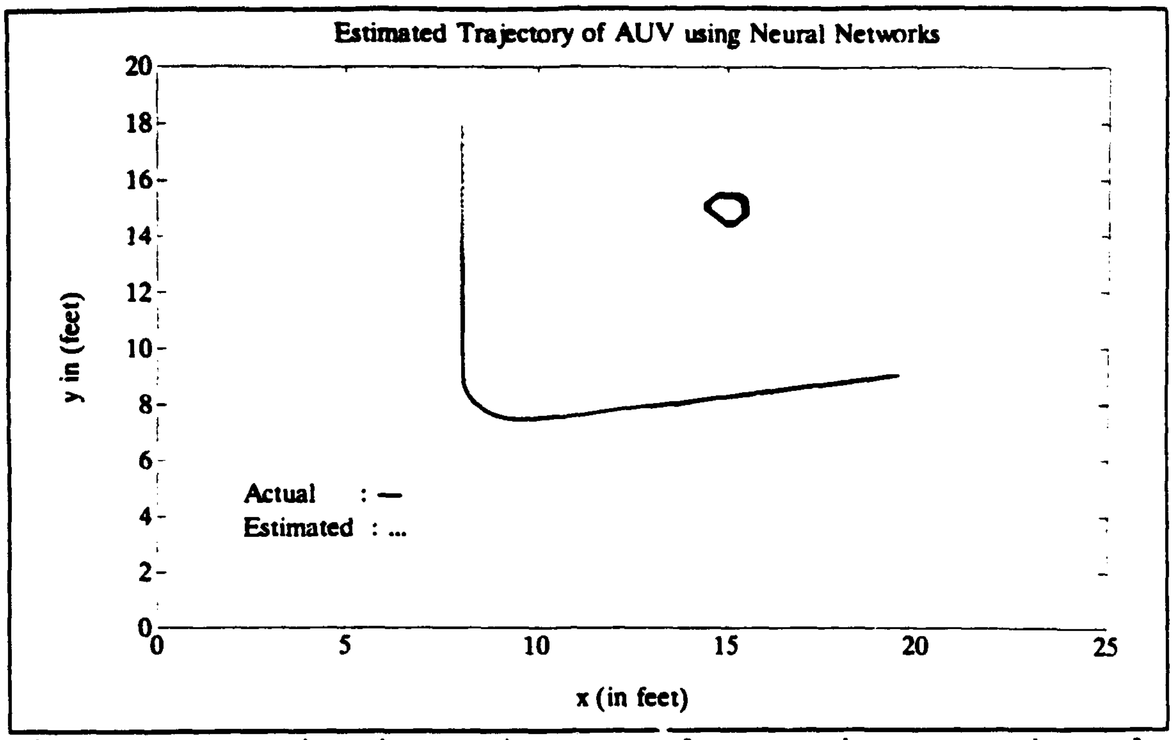

Figure 4.7 Estimation using neural networks, one obstacle present.

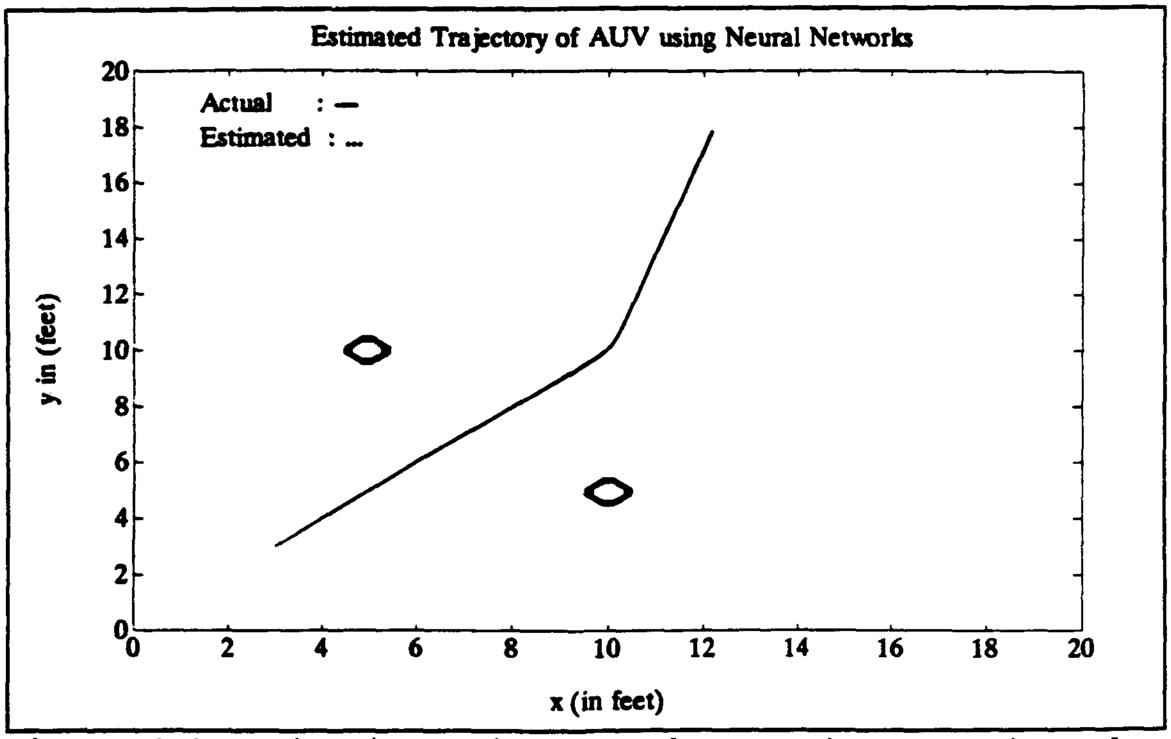

Figure 4.8 Estimation using neural networks, two obstacles present.

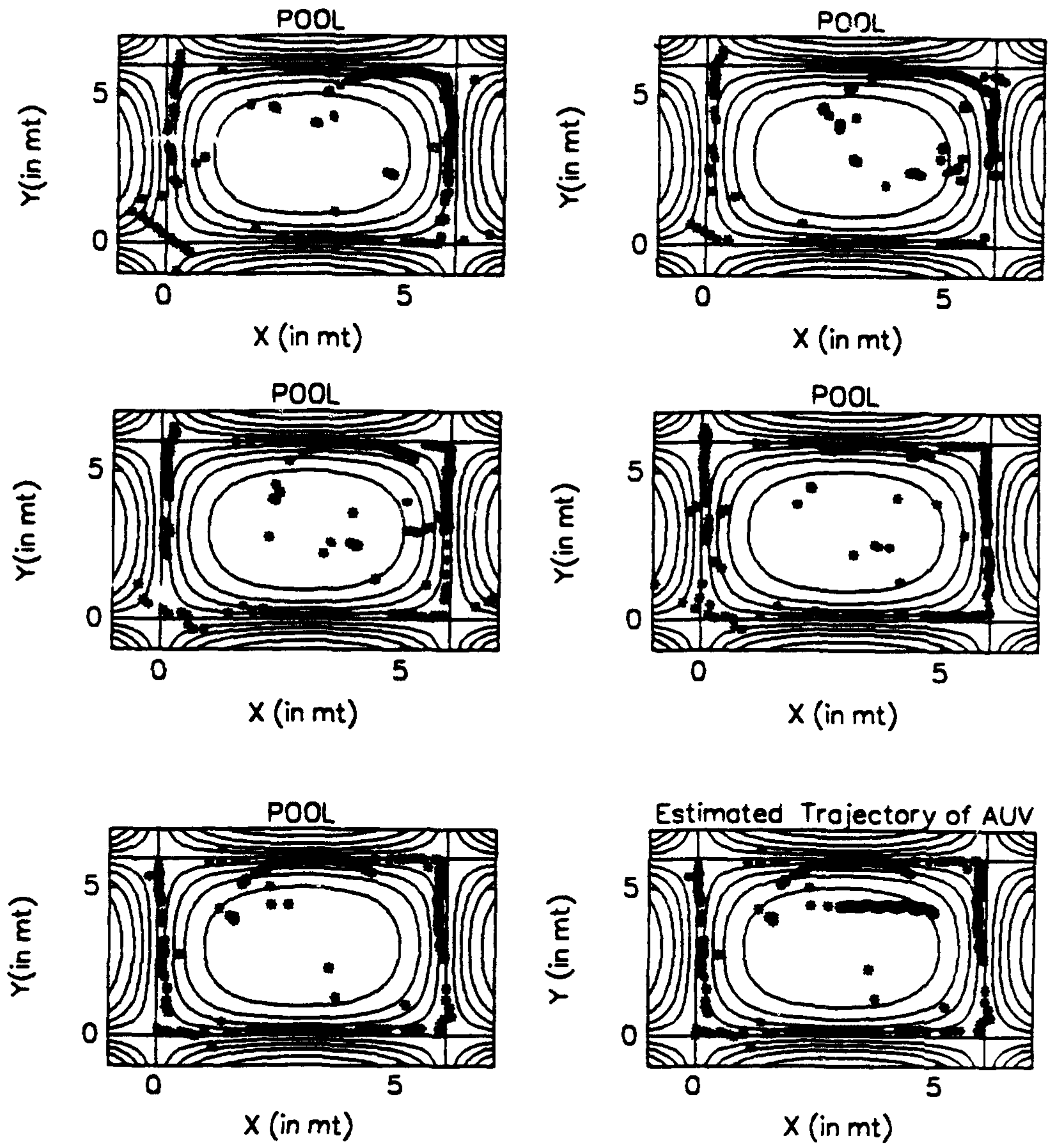

Figure 4.9 Estimation using neural networks with data collected in the water tank.

V. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

In this study the problem of navigating an Autonomous Underwater Vehicle in a partially known environment was investigated. In the navigation algorithm we made use of potential functions to define the area of operation. The potential function was combined with the vehicle dynamics in an extended Kalman filter. Throughout the simulation studies the algorithm proved to be robust even in the presence of obstacles that are not included in the map.

The method was first tested in a rectangular pool featuring several cylindrical buoys in the environment. Simulation studies were performed assuming known and unknown initial locations of the vehicle. Subsequently, the algorithm was applied in an open environment, such as the open sea, which included several obstacles. The obstacles that were in the area but which were not included in the map were identified by the algorithm. The effects of currents were also sought. Simulation studies showed that even in the presence of currents the algorithm gave a reasonable solution to the problem.

Finally, the algorithm was applied to the actual data collected in the water tank, The results with the experimental data proved that the algorithm could be reliably used in the implementation. A Fixed interval smoothing algorithm was also used in order to smooth the estimates produced by the Kalman filter. Throughout the simulation studies the dynamic parameters were estimated by least-squares techniques and these estimated parameters were used in the filter.

In Chapter IV a neural network was designed for use in the navigation algorithm. Basically, the potential function was replaced by a backpropagation neural network. This approach was successful only in the case of a known initial location.

The simulation studies using both computer generated and the actual data show that the navigation algorithm could be used in NPS AUVII. Several topics might further be investigated such as:

- Including more complex features in the potential function so that a more precise map of the area could be used;
- Training neural networks so that the neural network can be used in the presence of uncertain initial conditions; and
- Converting the navigator main program to C^{++} programming language for use in the NPS AUVII on board computer.

APPENDIX

A. PROGRAM STRUCTURE

The programming which covers an integral part of this study is accomplished using the software package. The main program is NAVIGATE.M. This program reads the data file, prompts the user for the values of the initial state estimates, error covariance and the parameter λ. There are three subroutine calls in this program. PAREST.M estimates the dynamic parameters associated with the system model using a least-squares algorithm. EST12P.M is the function that commands the estimator; it feeds the data to the Kalman filter, adjusts the parameter λ and does the plotting routines. SCAN10.M is the function that applies extended Kalman filtering. VP512.M is the function that calculates the potential function and its derivatives in the rectangular pool. SIGMOID.M and DSIG.M are the functions that gives the value of the sigmoid function and its derivative. SHOW10.M is the plotting routine. EST12P.M ,SCAN10.M ,SIGMOID.M, DSIG.M and SHOW10.M are modified from Ref[6].

VP_CYL.M is the function that gives the potential function for the rectangular pool when there is one cylindrical obstacle present. EST_OPNP.M is the function that commands the estimator when there are no borders. VP_OPEN.M and VP_OPEN2.M are the functions that give the potential function in an environment without borders. PLOT_OPEN2.M is the plotting routine for the potential function.

EST_CUR1.M and SCAN_CUR1.M are functions used to model the estimator in presence of currents. EST_ACT2.M and SCAN_ACT2.M are the functions that are used with actual data collected in the water tank. These functions also apply fixed interval smoothing to Kalman filter estimates. SMOOTH.M is the subroutine that accomplishes the fixed interval smoothing. This routine must be provided previously stored state estimates and predictions and associated error covariances.

TRAIN2.M is the M file that applies backprogation with adaptive learning rate and momentum using Neural Networks Toolbox of MATLAB. The program trains the network given input and target output sets and generates the weights and biases for the neuron layers. EST_NEURO.M and SCNNEURO.M are the functions that are used in neural network approach.

PR_DATA6.M and RANGEH.M are the M files modified from Ref[6] which are used to generate data for the simulations in the rectangular pool. PR_OPEN.M is the program to generate data for the simulations in an environment without borders.

B. PROGRAM LISTING

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%navigate.m
% Modified April 3rd 1994
% This program is the main program for the navigator
% The program prompts the user for the values of state and
%error
% covariance estimates and initial value of the parameter
%lambda.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i='Y'
while i=='Y'
      P=input('Enter initial state estimate:   ');
      zh0=input('Enter initial error covariance : ');
      lambda=input('Enter initial value of lambda :  ');
      % Estimate dynamic parameters.
      prms=parest(data);
      % Navigator algorithm
      zhm=est_act4(data,P,zh0,lambda,prms);
      % Continue with another simulation
      i=input('Do you want more simulations ? Y/N,'s');
      if isempty(i)
         i='Y'
      end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function zhm=est12p(data,P,zh0,L1,L2,lambda,prms)
% function zhm=est12p(data,P,zh0,L1,L2,lambda,prms)
% modified January 3rd 1994
% This program commands Kalman filter in the simulations for
%rectangular pool.
% zh0  is 5 x 1 matrix initial values of states
% P is  5X5 matrix  a  priori error covariance estimate.
% states=[x,y,v,heading,heading dot]
% L1 and L2 are the sides of the pool.
% prms  is dynamic parameters a1,a2,b1,b2
% data file has form:
% data=[alpha,rho,v,x,y,,rpm,rudder,theta,thetadot];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clg; hold off;clc
n=length(data);
```

```
axis('normal')
subplot(221)
zhm=zeros(5,1);
ns=400;

for i= 0:n/ns-1

[zh,P,zs]=scan10(zh0,P,data((ns*i+1):(ns*(i+1)),:),lambda,pr
ms,L1,L2);

     axis([-1 L1+1 -1 L2+1]);
     xm=zs(1,:); ym=zs(2,:);
     zhm=[zhm, zh];
     show10(xm,ym,lambda,L1,L2),title('POOL');
          if i==3
             meta alp1
          end
      hold off
     lambda=0.5*lambda;
      nz=length(zh); zh0=zh(:,nz);

end
hold off
show10(xm,ym,lambda,L1,L2);
hold on
plot(zhm(1,:), zhm(2,:),'*g'), title(' Estimated  Trajectory
of AUV')
meta
end   %est12p



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [zh,P,zs]=scan10(zh0,P,data,lambda,prms,L1,L2)
% modified 12 oct 93
% function [zh,P,zs]=scan10(zh0,P,data,lambda,prms,L1,L2)
% This function applies Kalman filter algorithm to the given
%data
% returned is the state estimates zh,estimated borders of
%the map and final error covariance.
% P 5 x 5 matrix,initial covarience matrix
% zh0 5 x 1 matrix initial values of states
% states=[x,y,v,heading,heading dot]'
% L1 and L2 are the sides of tthe rectangular pool.
% lambda initial parameter of potential function
% prms are the dynamic parameters a1,a2,b1,b2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

n=length(data);
zh(:,1)=zh0;
```

```
R=1;
a1=prms(1);a2=prms(2);b2=prms(3);b1=prms(4);
Ts=0.0225;
for t=1:n-1

    rho=data(t,2);
    alpha=data(t,1);
    headh=zh(4,t);
    RPM(t)=data(t,6);
    RUDDER(t)=data(t,7);
    a=cos(headh*pi/180);
    b=sin(headh*pi/180);

    A=[0 0 a 0 0
    0 0 b 0 0
    0 0 -a1 0 0
    0 0 0 0 1
    0 0 0 0 -b1];
    B=[0 0;0 0;a2 0;0 0;0 b2];
    [Phi,del]=c2d(A,B,Ts);
    dx0=rho*cos((alpha+headh)*pi/180);
    dy0=rho*sin((alpha+headh)*pi/180);
    x0=zh(1,t)+dx0;
    y0=zh(2,t)+dy0;
    zs(:,t)=[x0;y0];
    [v,dvx,dvy]=VP_cyl(x0,y0,lambda);
    h=[dvx,dvy,0,(-dvx*dy0+dvy*dx0)*pi/180,0]';
    s=h'*P*h+1; K=Phi*P*h/s;
    e=0-v;
    zh(:,t+1)=Phi*zh(:,t)+del*[RPM(t) RUDDER(t)]'+K*e;
    P=Phi*P*Phi'-K*s*K';

end
end   %scan

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function prms=parest(data);
% this function estimates the  dynamic parameters of the
% model using least squares estimates.
% the continous time parameters are returned.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%estimation for velocity parameters
a1=0;a2=0;a4=0;a5=0;a6=0;
v=data(:,3);
rpm=data(:,6);

for t=2:length(data);

    a1=a1+(v(t-1))^2;
    a2=a2+(v(t-1)*rpm(t-1));
    a4=a4+(rpm(t-1))^2;
    a5=a5+v(t-1)*v(t);
    a6=a6 + rpm(t-1)*v(t);

end
% discrete velocity parameters
thvelo=inv([a1 -a2;-a2 a4])*[-a5;a6];

% estimation for  heading parameters
thdot=data(:,9);
rudder=data(:,7);
b1=0;b2=0;b4=0;b5=0;b6=0;

for t=2:length(data);

    b1=b1+(thdot(t-1))^2;
    b2=b2+(thdot(t-1)*rudder(t-1));
    b4=b4+(rudder(t-1))^2;
    b5=b5+thdot(t-1)*thdot(t);
    b6=b6 + rudder(t-1)*thdot(t);
end
eps=1e-6
% discrete heading parameters.
thhead=inv([b1 -b2;-b2 b4])*[-b5;b6];
phi=[-thvelo(1) 0;0 -thhead(1)];
del=[thvelo(2) 0; 0 thhead(2)];
% discrete to continuous transformation
[A,B]=d2c(phi,del,0.0225);
prms=[-A(1,1) ;B(1,1);-A(2,2) ;B(2,2)];

end % parest



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [v,dvx,dvy]=VP512(x,y,lambda);
%[v,dvx,dvy]=VP(x,y,lambda)
% potential function for the pool.
% This function calculates potential function at a given
% point and its derivatives along x and y
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

v0=(x.*(x-12))*(y.*(y-12))';
z=(v0)/(lambda); v=sigmoid(z);
dvx=dsig(z)*((x-12)*(y.*(y-12))'+x*(y.*(y-12))')/lambda;
dvy=dsig(z)*((x.*(x-12))*(y-12)'+(x.*(x-12))*y')/lambda;
end  %potential
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=sigmoid(x)
% y =sigmoid(x)
% value of sigmoid function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x=min(x,100); x=max(x,-100);
y= (1-exp(-x))./(1+exp(-x));
end % sigmoid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function d=dsig(x)
%derivative of sigmoid
% d=dsig(x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x=min(x,100);x=max(x,-100);
temp=exp(-x);
d=temp ./ (1+2*temp + temp .* temp);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x=show10(xm,ym,lambda,L1,L2)
%plottig routine for displaying the  estimated borders of
%the map and contours of potential function.
% clg; hold off;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x=linspace(-1,L1+1);
y=linspace(-1,L2+1);
[v,dvx,dvy]=VP_cyl(x',y',lambda);
contour(v,x,y)
hold on
for t=1:length(xm)
plot(xm(t), ym(t), 'og'),xlabel('X (in feet)');
ylabel('Y(in feet)'),
end
end %show





%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [v,dvx,dvy]=vp_cyl(x,y,lambda);
%[v,dvx,dvy]=VP_cyl(x,y,lambda)
% potential function for the pool.
% this function defines a cylindrical buoy in the pool
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
L1=12;L2=12;
a=9;b=9;c=.04;  % radius of the object is .2 ft.
v0=(x.*(x-L1).*(x-a).^2)*(y.*(y-L2))'+(x.*(x-L1))*(y.*(y-L2)
...*(y-b).^2)'-c*(x.*(x-L1))*(y.*(y-L2))';
z=(v0)/(lambda); v=sigmoid(z);  v=flipud(v);
dvx=dsig(z)*(((x-L1).*(x-a).^2)*(y.*(y-L2))'+(x.*(x-a).^2)
*(y.*(y-L2))'+(2*x.*(x-L1).*(x-a))*(y.*(y-L2))'...
+(x-L1)*(y.*(y-L2).*(y-b).^2)'+x*(y.*(y-L2).*(y-b).^2)'...
-c*(x-L1)*(y.*(y-L2))'-c*x*(y.*(y-L2))')/lambda;
dvy=dsig(z)*((x.*(x-L1).*(x-a).^2)*(y-L2)'...
+(x.*(x-L1).*(x-a).^2)*y'+(x.*(x-L1))*((y-L2).*(y-b).^2)'...
+(x.*(x-L1))*(y.*(y-b).^2)'+(x.*(x-L1))*(2*y.*(y-L2).*(y-b))
...'-c*(x.*(x-L1))*(y-L2)'-c*(x.*(x-L1))*y')/lambda;

end       %vp_cyl


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [zhm]=est_opnp(data,P,zh0,lambda,prms)
% function zhm=est_opnp(data,P,zh0,lambda,prms)
% modified  April 1 st 1994
% this function commands estimator for the case of open sea
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clg; clc;
n=length(data);
ns=200;
for i= 0:n/ns-1

[zh,P,zs]=scn_opn(zh0,P,data((ns*i+1):(ns*(i+1)),:),lambda,p
rms);
    zhm=[zhm, zh];
    zsm=[zsm,zs];
    lambda=.3*lambda
    nz=length(zh); zh0=zh(:,nz);

end
clg
lambda,pause
axis([0 25 -5 20])
plt_opn2(25,15);
hold on
plot(zhm(1,:), zhm(2,:),'*g',data(:,4),data(:,5),'-.'),
title(' Estimated  Trajectory of AUV'),grid
xlabel(' X (in feet)'),ylabel( 'Y (in feet)'),
gtext('Estimated :Solid  line'),
gtext('Actual    :Dashed line'),
hold off
end     % est_opnp
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  [v ,dvx,dvy]=vp_open(x,y,lambda);
%modified February 19th 1994
%This function calculates the potential function
% in a environment without borders
%[v,dvx,dvy]=vp_open(x,y,lambda);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a=5; b=5; c=.25;
v0=(x-a).^2+(y-b).^2-c;
z=v0/lambda; v=sigmoid(z);,
dvx=dsig(z)*2*(x-a)/lambda;
dvy=dsig(z)*2*(x-b)/lambda;
end     % vp_open


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  [v ,dvx, dvy]=vp_open2(x,y,lambda);
%modified January 20th 1994.
% vp_open2.m
%function for computing potential function for two obstacles
% in open sea.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a=10; b=20; c=.25;d=16;e=10;f=.25;
v0=((x-a).^2)*((x-d).^2)+((y-b).^2)*((x-d).^2)-(c*(x-d).^2)+
((x-a).^2)*(y-e).^2+((y-b).^2)*(y-e).^2-c*(y-e).^2-...
f*(x-a).^2-f*(y-b).^2+c*f;
z=v0/lambda; v=sigmoid(z);,
dvxx=2*(x-a)*((x-d).^2)+2*((x-a).^2)*(x-d)+...
2*(x-d)*((y-b).^2)-2*c*(x-d)+2*(x-a)*((y-e).^2)-2*f*(x-a);
dvx=dsig(z)*dvxx/lambda;
dvyy=2*(y-b)*((x-d).^2)+2*(y-e)*((x-a).^2)+2*(y-b)*...
((y-e).^2)+2*(y-e)*((y-b).^2)-2*c*(y-e)-2*f*(y-b);
dvy=dsig(z)*dvyy/lambda ;
end          % vp_open2


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function   plt_opn2(limits,lambda)
% routine for plotting potential function
%  when two obstacles are present.
%modified January 20th 1994
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xx=linspace(0,limits);,yy=linspace(0,limits);
[x, y]=meshdom(xx,yy);
a=10;b=20;c=.25;d=16;e=10;f=.25;
u=((x-a).^2+(y-b).^2-c).*((x-d).^2+(y-e).^2-f)/lambda;
z=(1-exp(-u))./(1+exp(-u)); ,% mesh(z),pause
%title('Potential Function When Two Obstacles are present')
%meta alp1
contour(z,xx,yy);
end  % plt_opn2
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n zhm=est_curl(data,P,zh0,lambda,prms)
% function zhm=est_curl(data,P,zh0,lambda,prms)
% modified 21 Nov 1993
% this function commands the estimator for the case of
%currents
% states=[x, y, v, theta,thetadot ,currentx,currenty] 7X1
vector
% P=7X7 error covariance vector.
% zh0= initial state estimates 7X1 vector.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clg; hold off;clc
n=length(data);
ns=200;

for i= 0:n/ns-1

[zh,P,zs]=scn_curl(zh0,P,data((ns*i+1):(ns*(i+1)),:),lambda,
prms)
    zhm=[zhm, zh];
    lambda=.5*lambda
    nz=length(zh); zh0=zh(:,nz);

end

clg
axis([0 20 0 25])
plt_opn2(30,1)
hold on
plot(zhm(1,:), zhm(2,:),'*g',data(:,4),data(:,5),'-.')
title(' Estimated  Trajectory of AUV'),grid
xlabel('X (in feet)'),ylabel('Y (in feet)')
gtext('Estimated :Solid  line') ,
gtext('Actual    :Dashed line'),

end    % est_curl




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [zh,P,zs]=scn_curl(zh0,P,data,lambda,prms)
% modified April 2nd 1994
% this function applies Kalman filter when there are
% currents present
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
n=length(data);
zh(:,1)=zh0;
R=1; %Measurement error Covariance.
a1=prms(1);b1=prms(2);b2=prms(3);b1=prms(4);Ts=0.0225;
for t=1:n-1

    rho=data(t,2);
    alpha=data(t,1);
    headh=zh(4,t);
    RPM(t)=data(t,6);
    RUDDER(t)=data(t,7);
    a=cos(headh*pi/180);
    b=sin(headh*pi/180);
    A=[0 0 a 0 0 1 0;
    0 0 b 0 0 0 1;
    0 0 -a1 0 0 0 0;
    0 0 0 0 1 0 0;
    0 0 0 0 -b1 0 0;
    0 0 0 0 0 0 0;
    0 0 0 0 0 0 0];
    B=[0 0;0 0;a2 0;0 0;0 b2;0 0;0 0];
    [Phi,del]=c2d(A,B,Ts);
    dx0=rho*cos((theta+headh)*pi/180);
    dy0=rho*sin((theta+headh)*pi/180);
    x0=zh(1,t)+dx0;
    y0=zh(2,t)+dy0;
    zs(:,t)=[x0;y0];
    [v,dvx,dvy]=vp_open2(x0,y0,lambda);
    h=[dvx,dvy,0,(-dvx*dy0+dvy*dx0)*pi/180,0,0,0]';
    s=h'*P*h+R; K=Phi*P*h/s;
    e=0-v;
    zh(:,t+1)=Phi*zh(:,t)+del*[RPM(t) RUDDER(t)]'+K*e;
    P=Phi*P*Phi'-K*s*K';

end

end % scan_curl




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  [zhm,zsmt] = est_act2(data,Pk1k,zh0,lambda)
%[zhm,zsmt]=est_act2(data,P,zh0,lambda)
% Modified February 3 1994
% data file must contain:[bearing(relative),range]
% estimated states are zhm=[x, y,v,theta]
% all angles are in degrees ranges are in meters.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
! del tezg1.met
clg; hold off;
subplot(221)
n=length(data);
ns=200;
for i=0:n/200-1

[zh,Pk1k,zs,xsmth]=scn_act2(zh0,Pk1k,data((ns*i+1):(ns*(i+1)
),:),lambda);
    xm=zs(1,:); ym=zs(2,:);
    zhm=[zhm, zh];
    zsmt=[zsmt,xsmth];
    show11(xm,ym,lambda); title('POOL');
    if i==3
       meta
    end
    hold off
    lambda=0.5*lambda;
    nz=length(zh); zh0=zh(:,nz);

end
hold off
show10(xm,ym,lambda);
hold on
plot(zhm(1,:), zhm(2,:),'*g'), title(' Estimated  Trajectory
of
AUV')
hold off
show10(xm,ym,lambda);
plot(zsmt(1,:),zsmt(2,:),'*r'),title(' Smoothed Trajectory
of AUV'),
hold off
meta
end % est_act2




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [zh,Pk1k,zs,xsmth]=scn_act2(zh0,Pk1k,data,lambda)
% [zh,Pk1k,zs,xsmth]=scn_act2(zh0,Pk1k,data,lambda)
% Modified February 3rd 1994
% The estimation algorithm for navigation of AUV.
% This function works with actual data taken at Pool.
% Fixed interval smoothing is applied to the estimated
%states
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=length(data);
zh(:,1)=zh0;
```

```
dt=0.0225;
R=1;
for t=1:n-1

    headh=zh(4,t);
    rho=data(t,2); theta=data(t,1);
    A=[0 0 cos(headh*pi/180) 0;
    0 0 sin(headh*pi/180) 0;
    0 0 0 0;
    0 0 0 0];
    B=[0 0 0 0]';
    [Phi del]=c2d(A,B,dt);
    dx0=rho*cos((theta+headh)*pi/180);
    dy0=rho*sin((theta+headh)*pi/180);
    x0=zh(1,t)+dx0;
    y0=zh(2,t)+dy0;
    zs(:,t)=[x0;y0];
    [v,dvx,dvy]=VP(x0,y0,lambda);
    %Kalman filter
    H=[dvx,dvy,0,(-dvx*dy0+dvy*dx0)*pi/180];
    KG=Pk1k*H'*(H*Pk1k*H'+R)^(-1);
    Pkk=(eye(4)-KG*H)*Pk1k;
    Pkkc=[Pkkc Pkk];
    Pk1k=Phi*Pkk*Phi';
    Pk1kc=[Pk1kc Pk1k];
    zh1(:,t)=zh(:,t)+KG*(0-v);
    zh1c(:,t)=zh1(:,t);
    zh(:,t+1)=Phi*zh1(:,t);
    zhc=[zhc,zh(:,t+1)];

end

% smooth data points
[xsmth]=smooth(zhc,zh1c,Pkkc,Pk1kc,data);

end %scn_act



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 function [xsmth]=smooth(zhc,zh1c,Pkkc,Pk1kc,data);
% function [xsmth]=smooth(zhc,zh1c,Pkkc,Pk1kc,data);
%  Modified February 11th 1994;
% This Subroutine implements the Fixed interval smoothing
% to the previously stored error covariance and state
%estimates.
%This function works in reverse time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Flip the Previously stored Matrices
Pkkc=fliplr(Pkkc);
```

```
Pk1kc=fliplr(Pk1kc);
zh1c=fliplr(zh1c);
zhc=fliplr(zhc);
%initial smoothed estimate is final  estimate zh;
xs(:,1)=zhc(:,1);
%initial smoothed error covariance is final forward time
% error covariance
Ps1=fliplr(Pkkc(:,1:4));

for i =1:length(data)-5;
    headh=xs(4,i);
    A=[0 0 cos(headh*pi/180) 0;
    0 0 sin(headh*pi/180) 0;
    0 0 0 0;
    0 0 0 0];
    B=[0 0 0 0]';
    [Phi,del]=c2d(A,B,dt);
% Smoothing Filter.
Ak=fliplr(Pkkc(:,4*i-3:4*i))*Phi'*...
(fliplr(Pk1kc(:,4*i-3:4*i)))^(-1);
xs(:,i+1)=zh1c(:,i) + Ak*(xs(:,i)-zhc(:,i));
Ps1=fliplr(Pkkc(:,4*i-3:4*i))+...
Ak*(Ps1-fliplr(Pk1kc(:,4*i-3:4*i)))*Ak';

end

% Flip the smoothed estimates to forward time
xsmth=fliplr(xs);
end   % smooth
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Train2.m
%modified December 17th 1993
%This m file applies  momentum with adaptive learning rate
%algorithm to train input vectors.
% Input vector = [x0,y0] location of the sonar tip
% Output target vector =[v dvx dvy] potential function and
%its  derivatives
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
%load nrin2.dat
```

```
%load nrout2.dat

P=nrin2
T=nrout2;
% initialize network
[R,Q]=size(P)
S1=9;
[S2,Q]=size(T)
[W1,B1]=nwtan(S1,R)
[W2,B2]=rands(S2,S1)
%Training parameters

disp_freq=25            % display frequency
max_epoch=400;          % maximum iterations
err_goal=.002;          % error level
momentum=.95;           % amount of momentum
err_ratio=1.04;         % error ratio
lr=0.5                  %learning ratio
lr_inc=1.05             %increment
lr_dec=.7               %decrement

%training parameters
TP=[disp_freq max_epoch err_goal lr  lr_inc lr_dec  momentum
err_ratio];
% Train Network

[W1,B1,W2,B2,epochs,TR]=trainbpx(W1,B1,'tansig',W2,B2,'purel
in',P,T,TP);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function zhm=estneuro(data,P,zh0,prms,Wa,Wb,Ba,Bb)
% function zhm=estneuro(data,P,zh0,prms,Wa,Wb,Ba,Bb)
% modified 11 Nov 1993
%this function implements neural networks for estimation.
% Wa,Ba are weights and biases for first (sigmoid) layer
% Wb,Bb are weights and biases for second (linear) layer
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clg; hold off;clc
subplot(221)
n=length(data);
Ts=0.0225;
```

```
[zh,P,zs]=scnneuro(zh0,P,data,prms,Wa,Wb,Ba,Bb);

axis([0 20  0 25])
plt_opn2(25,.1 ),hold on,
plot(zhm(1,:), zhm(2,:),'*g',data(:,4),data(:,5),'-.'),
title(' Estimated  Trajectory of AUV using BNN'),grid
xlabel(' X(in feet)'),ylabel(' Y(in feet)'),
gtext('Estimated :Solid  line') ,
gtext('Actual    :Dashed line'),
end     % estneuro




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [zh,P,zs]=scnneuro(zh0,P,data,prms,Wa,Wb,Ba,Bb)
%modified  November 11th 1993
% this function applies the estimation algorithm using
% back propagation neural networks
%potential function is replaced by neural network.
% Wa,Ba are weight and biases for first (sigmoid ) layer
% Wb, Bb are weight and biases for second (linear) layer
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=length(data);
zh(:,1)=zh0;
R=1; %Measurement error Covariance.
a1=prms(1);a2=prms(2);b2=prms(3);b1=prms(4);Ts=0.0225;
for t=1:n-1

    rho=data(t,2);
    alpha=data(t,1);
    headh=zh(4,t);
    RPM(t)=data(t,6);
    RUDDER(t)=data(t,7);
    a=cos(headh*pi/180);
    b=sin(headh*pi/180);
    A=[0 0 a 0 0
    0 0 b 0 0
    0 0 -a1 0 0
    0 0 0 0 1
    0 0 0 0 -b1];
    B=[0 0;0 0;a2 0;0 0;0 b2];
    [Phi,del]=c2d(A,B,Ts);
    dx0=rho*cos((alpha+headh)*pi/180);
    dy0=rho*sin((alpha+headh)*pi/180);
    x0=zh(1,t)+dx0;
    y0=zh(2,t)+dy0;
    zs(:,t)=[x0;y0];
    % implementation of neural network.
    Aa=tansig((Wa*[x0;y0]),Ba);      % sigmoid neuron layer
    Ab=purelin(Wb*Aa,Bb);                 % linear  neuro  layer
    v=Ab(1);dvx=Ab(2);dvy=Ab(3);          % network output
    h=[dvx,dvy,0,(-dvx*dy0+dvy*dx0)*pi/180,0]';
    s=h'*P*h+1; K=Phi*P*h/s;
    e=0-v;
    zh(:,t+1)=Phi*zh(:,t)+del*[RPM(t) RUDDER(t)]'+K*e;
    P=Phi*P*Phi'-K*s*K';

end

end %scnneuro




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PRDATA6.M
% modified November 18 1993
%this program simulates the motion of AUV in the rectangular
%pool
% several obstacles are added in the environment.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear,clg,clc
L1=12;L2=12;
Ts=0.0225;
Tf=7*9;
kmax=Tf/Ts+1;
x=zeros(5,kmax);
x(:,1)=[2 4 3 0 0]';
time(1)=0;
RPM=150*ones(1,kmax);
RUDDER=zeros(1,kmax);
%dynamic parameters
a1=1;
a2=0.001;
b2=1;
b1=1;
for k=1:kmax-1

    time(k+1)=time(k)+Ts;
    tetha=x(4,k);
    a=cos(tetha*pi/180);
    b=sin(tetha*pi/180);
    A=[0 0 a 0 0
       0 0 b 0 0
       0 0 -a1 0 0
       0 0 0 0 1
       0 0 0 0 -b1];
    B=[0 0;0 0;a2 0;0 0;0 b2];
    E=eye(5);
    [phi,del1]=c2d(A,B,Ts);
    [phi,del2]=c2d(A,E,Ts);
    if time(k)>=10
       RUDDER(k)=3;
    end
    if time(k)>=35
       RUDDER(k)= 0;
    end

    rand('normal')
    ex=0.01*rand;
    ey=0.01*rand;
    ev=0.01*rand;
    et=0.01*rand;
    etd=0.01*rand;
    x(:,k+1)=phi*x(:,k)+del1*[RPM(k) RUDDER(k)]'+del2*[ex ey
ev
et etd]';
    head(k)=x(4,k);
    shdg(k)=rem(head(k)+0.9*k,360);
    shdg1(k)=rem(0.9*k,360);
    if shdg(k) > 180
       shdg(k)=-360+shdg(k);
    end
    if shdg1(k) > 180
       shdg1(k)=-360+shdg1(k)+0.01*rand;
    end
    ang1(k)=180/pi*atan2((3-x(2,k)),(9-x(1,k)));  % obstacle
at(9,3)
    ang2(k)=180/pi*atan2((9-x(2,k)),(3-x(1,k)));  % obstacle
at(3,9)
    [r,h]=rangeh(x(1,k),x(2,k),shdg(k),L1,L2);

    if (shdg(k)>=ang1(k)-3)&(shdg(k)<=ang1(k)+3);
       dist(k)=sqrt((9-x(1,k))^2+(3-x(2,k))^2)+0.01*rand;
    elseif (shdg(k)>=ang2(k)-2)&(shdg(k)<=ang2(k)+2);
       dist(k)=sqrt((3-x(1,k))^2+(9-x(2,k))^2)+0.01*rand;
    else
       dist(k)=r+0.01*rand;
    end
 end
    temp(k,:)=[shdg1(k) dist(k) x(3,k) x(1,k) x(2,k) RPM(k)
RUDDER(k)...
                   x(4,k) x(5,k)];
end
!del data1p5.dat
% save data in ascii code
save data1p5.dat temp /ascii
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [r,h]=rangeh(x,y,theta,L1,L2)
% [r,h]=range(x,y,theta,L1,L2)
% this function computes sonar range rho
% and its gradient h
% at position (x,y) with heading theta
% in the L1 by L2 pool.
% h=[dr/dtheta, dr/dx, dr/dy]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

theta=theta*pi/180;
th1=atan2(-y,-x);
th2=atan2(L2-y,-x);
th3=atan2(L2-y,L1-x);
th4=atan2(-y,L1-x);
while (theta >th1+2*pi),
    theta=theta-2*pi;
end
while (theta <=th4),
    theta=theta+2*pi;
end
c=cos(theta);s=sin(theta);
if (theta>=th2)&(theta<th1+2*pi),
     r=-x/c;
     h=[-x*s/c^2,-1/c,0];
end
if (theta>=th3)&(theta<th2),
     r=(L2-y)/s;
     h=[(y-L2)*c/s^2,0,-1/s];
end
if (theta>=th4)&(theta<th3),
     r=(L1-x)/c;
     h=[(L1-x)*s/c^2,-1/c,0];
end
if (theta>=th1+2*pi)&(theta<th4+2*pi),
     r=-y/s;
     h=[-y*c/s^2,0,-1/s];
end
end %range


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%pr_open.m
%This program simulates the motion of AUV
%in a environment without borders
%modified 25 oct 93
%several obstacles in the environment can be added.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear,clg,clc
```

```
Ts=0.0225;
Tf=7*9;
kmax=Tf/Ts+1;
x=zeros(5,kmax);
x(:,1)=[3 3 3 45 0 ]';
time(1)=0;
RPM=150*ones(1,kmax);
RUDDER=zeros(1,kmax);
a1=1;
a2=0.001;
b1=1;
b2=1;
for k=1:kmax-1

     time(k+1)=time(k)+Ts;
     tetha=x(4,k);
     a=cos(tetha*pi/180);
     b=sin(tetha*pi/180);
     A=[0 0 a 0 0
        0 0 b 0 0
        0 0 -a1 0 0
        0 0 0 0 1
        0 0 0 0 -b1];
     B=[0 0;0 0;a2 0;0 0;0 b2];
     E=eye(5);
     [phi,del1]=c2d(A,B,Ts);
     [phi,del2]=c2d(A,E,Ts);

if time(k)>=30;          ;
   RUDDER(k)=10;
end
if time (k) >=33
   RUDDER(k)=0;
end
     rand('normal')
     W=0.01*rand*[1 1 1 1 1]'
         x(:,k+1)=phi*x(:,k)+del1*[RPM(k) RUDDER(k)]'+del2*W;
     head(k)=x(4,k);
     shdg(k)=rem(head(k)+0.9*k,360);
     shdg1(k)=rem(0.9*k,360);
     if shdg(k) > 180
        shdg(k)=-360+shdg(k);
     end
     if shdg1(k) > 180
        shdg1(k)=-360+shdg1(k)+0.01*rand;
     end
     ang1(k)=180/pi*atan2((10-x(2,k)),(5-x(1,k)));        %1st
obstacle at (5,10)
     ang2(k)=180/pi*atan2((5-x(2,k)),(10-x(1,k)));%    2nd
obstacle at (10,5)
```

```
    ang3(k)=180/pi*atan2((5-x(2,k)),(18-x(1,k)));    %    3rd
obstacle at (18,5)
    ang4(k)=180/pi*atan2((10-x(2,k)),(2-x(1,k)));%    4th
obstacle at (2,10)
    ang5(k)=180/pi*atan2((20-x(2,k)),(8-x(1,k)));%    5th
obstacle at (8,20

   if (shdg(k)>=ang1(k)-2)&(shdg(k)<=ang1(k)+2);
       dist(k)=sqrt((5-x(1,k))^2+(10-x(2,k))^2)+0.01*rand;
   elseif (shdg(k)>=ang2(k)-2)&(shdg(k)<=ang2(k)+2);
       dist(k)=sqrt((10-x(1,k))^2+(5-x(2,k))^2)+0.01*rand;
   elseif (shdg(k)>=ang3(k)-2)&(shdg(k)<=ang3(k)+2);
       dist(k)=sqrt((18-x(1,k))^2+(5-x(2,k))^2)+0.01*rand;
   elseif (shdg(k)>=ang4(k)-2)&(shdg(k)<=ang4(k)+2);
      dist(k)=sqrt((2-x(1,k))^2+(10-x(2,k))^2)+0.01*rand;
   elseif (shdg(k)>=ang5(k)-2)&(shdg(k)<=ang5(k)+2);
       dist(k)=sqrt((8-x(1,k))^2+(20-x(2,k))^2)+0.01*rand;
   else
       dist(k)=500+0.01*rand; % add very large number
   end
    temp(k,:)=[shdg1(k) dist(k) x(3,k) x(1,k) x(2,k) RPM(k)
RUDDER(k) x(4,k) x(5,k)];
end
del datalp1.dat
% save data in ascii format in a data file
save datfeb1.dat temp /ascii
```

LIST OF REFERENCES

1. Friend, John R., *Design of a Navigator for a Testbed Autonomous Underwater Vehicle*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1989.
2. Zyda, Michael, and others, " Three Dimensional Visualization of Mission Planning and Control for the NPS Autonomous Underwater Vehicle," *IEEE Journal of Oceanic Engineering*, Vol 15, No 3, p. 217, July 1990.
3. Burl, J., B., " Derivation of Kalman Filter Equations," notes for the EC 3310 (Linear Optimal Estimation and Control), Naval Postgraduate School, 1990, (unpublished).
4. Meditch, J. S., *On Optimal Linear Smoothing Theory,* Technical Report 67-105, Information Processing and Control Systems Laboratory, Evanston, Illinois, March 1967.
5. Ljung, L., *System Identification Theory for the User,* Prentice Hall Information and System Sciences series, 1987.
6. Percin, E., *Sonar Localization of an Autonomous Underwater Vehicle*, Master's Thesis, Naval Postgraduate School, Monterey, California, 1993.
7. Cristi, R., "Sensor Based Navigation of an Autonomous Underwater Vehicle," paper presented at the International Symposium on Unmanned, Untethered Submersible Technology, Durham, New Hampshire, August, 1993.
8. Beale, M. and Demuth, H., *Neural Networks Toolbox*, The Mathworks Inc, 1992.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library Code 52 Naval Postgraduate School Monterey, CA 93943-5101	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
4.	Professor R. Cristi, Code EC/Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
5.	Professor H. Titus, Code EC/Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
6.	Deniz Kuvvetleri Komutanligi Personel Daire Baskanligi Bakanliklar, Ankara, TURKEY	1
7.	Deniz Harp Okulu Komutanligi Tuzla, Istanbul, TURKEY	2
8.	Golcuk Tersanesi Komutanligi Golcuk, Kocaeli, TURKEY	1
9.	Taskizak Tersanesi Komutanligi Taskizak, Istanbul, TURKEY	1
10.	Alp Kayirhan Lise Cad 92/4 Denizli, TURKEY	1
11.	Ziya Kayirhan Lise Cad 92/4 Denizli, TURKEY	1